

Capturing Political Communication Online Using Image and Text Data: A Deep Learning Approach

by


Alejandro Javier Pineda

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Political Science and Scientific Computing)
in The University of Michigan
2023

Doctoral Committee:

Professor Walter Mebane, Chair
Assistant Professor Christopher Fariss
Professor Vincent Hutchings
Assistant Professor Mara Ostfeld
Associate Professor Josh Pasek

© Alejandro Pineda 2023
All Rights Reserved

Alejandro Pineda 
ajpineda@umich.edu
ORCID iD: 0000-0003-0162-3042

For Kilo – we did it, buddy.
Let's go for a walk.

ACKNOWLEDGEMENTS

In my imagination, I have written this page a thousand times. On long walks with my dog, Kilo, I'd imagine what it would feel like to reach *this* point in my doctoral program – not quite at the mountain top, but close enough to enjoy the view. At present, I am sitting in a café in downtown Ypsi, staring at the library across the way. As I practice ordering a chocolate chip cookie in my head, I can't help but notice the trees in Michigan have mostly lost their leaves. I still have revisions to make, but with each hour that passes, I grow painfully aware that this journey is coming to an end. After seven and a half years, I am a short climb from the summit.

Gods, what a marvelous view.

I am nothing if not the product of excellent teachers. I have had so many over the years – both in and out of the classroom – that listing them all here, one by one, seems excessive. I will do my best to highlight the most significant individuals as concisely as I can.

First, I have to thank my most excellent committee: Professors Mara Ostfeld, Christopher Fariss, Vincent Hutchings, Walter Mebane, and Josh Pasek. As academics at the University of Michigan, each of them represent the best of their respective fields. Across political science and communication, they have produced scholarship of the finest caliber. I hope this dissertation rises to a similar degree.

Second, I'd like to thank the group of professors I had as an undergraduate, at Santa Clara University, who first inspired me to pursue a doctorate. Thank you to Professors Timothy Lukes, Peter Minowitz, Jim Cottrill, Bill Sundstrom, and Naomi Levy. Collectively, you were the Gandalf to my Bilbo – pushing me out of my quiet little Hobbit dwelling with the assurance that everything will be okay.

A third wave of thanks to the teachers at Junipero Serra High School and Saint Gregory Elementary. Especially Mr. Morton, Mr. Ferrando, and Mr. Casey (I'd address you by your first names, but somehow, it would feel disrespectful). It was in these classrooms where I fell in love with learning.

There is a group of guys known collectively as The Gentlemen’s Fantasy League¹...I suppose I’ll have to thank them next. Gentlemen: may all of your fantasy football teams lose this weekend. Except you, Sean Smith – you’re perfect. Thank you gentlemen for the endless loyalty, love, and support. Perhaps the greatest cost I had to pay to complete this work was moving to Michigan, away from you all. By that same token, the greatest support I had was knowing that you idiots were always in my corner, even if I failed.

Speaking of people in your corner: thank you to my coaches and teammates in the boxing community. Thank you to the University of Michigan Boxing Team, A-Square Fight Club, and Jabs Gym. Thank you to my coaches Tony Sensoli, Akio Miller, Eric McGuire, and Joe Luis Posada. Thank you to Reggie Harris and Jasmine Hampton for being constant sources of inspiration. Thank you to Rama for giving me the most useful advice that one time: “don’t you *dare* quit on me, Alejandro.” Without the discipline, work ethic, and humility I learned from the sport of boxing, I simply do not know how this dissertation gets written.

Thank you to the employees of: Oz Cannabis, Dos Hermanos Market, Lan City Noodles, Family’s Fried Chicken, Abe’s Coney Island. Thank you to Alice Kelley, Regan Elizabeth, and Kasey Bradford; thank you to Tori Noe and David Oblak. Thank you to my *awesome* team at VMLY&R: Chris Lundquist, Rich Rangel, Poli Changdar, and Bill Herrmann. Thank you to my haters and doubters for the fuel (gods bless you, you are doing the Lord’s work).

Thank you, Ypsilanti! Speaking of Ypsi: if there are any seeds of revolution within these pages, I assure you, they were found in the flowers at Bridge Community Café. I must be joining a long tradition of scholars indebted to coffee shop owners. Thank you to Sierra and the entire team there. I am so grateful – not only for the espresso, pastries, and savory pies – but also for the positive energy. The smiles radiating from this little spot in downtown fill this weary scholar with such hope.

Finally – and most significantly – thank you to my family. To my mother, Belkis, who gave me life, and then taught me to live with strength and grace. To my father, Rafael, who passed many years ago, but whose voice echoes in the trees. And to my sisters, Christina and Laura: thank you for being the two *coolest* teachers.

Alright, nerds – let’s do some political science, shall we?

-ajp

¹Current gentlemen include: Paul, Max, Peter, Jake, John, Quin, Dane, Brett, Tyler, Nate, Ryan, Lauterbach, & McMinn

TABLE OF CONTENTS

DEDICATION	ii
ACKNOWLEDGEMENTS	iii
LIST OF FIGURES	vii
CHAPTER	
I. Introduction: Of Politics and Computers	1
II. Algorithmic Bias in Deep Learning: An Inquiry Into Race Recognition	3
2.1 Introduction	4
2.2 Fairness Problems and Sampling Solutions	6
2.2.1 Machine Learning in Political Communication	7
2.3 Deep Learning for Race Classification	7
2.3.1 Transfer Learning	8
2.4 Image Processing and Experiment Design	11
2.4.1 Experiment Design	14
2.5 Results and Discussion	16
2.5.1 Discussion	20
2.6 Appendix I: Code Overview	24
III. Multimodal Deep Learning for Detecting Election Adminis- tration Issues on Social Media	29
3.1 Introduction	30
3.2 Election Administration Issues and Their Impact	32
3.3 Twitter Scraping During the 2016 U.S. Election	35
3.4 Deep Learning for Tweet Classification	37
3.4.1 Transfer Learning	38
3.5 Results and Discussion	42
3.6 Appendix I: Code Overview	51

IV. Multimodal Deep Learning for Capturing Attitudes Toward Black Lives Matter	57
4.1 Introduction	58
4.2 Capturing Public Opinion Toward BLM Protests	59
4.3 Collecting & Coding Black Lives Matter Tweets	63
4.4 Multimodal Deep Learning for Tweet Classification	68
4.4.1 Transfer Learning	69
4.5 Results and Discussion	74
4.6 Appendix I: Code Overview	80
V. Conclusion	87
BIBLIOGRAPHY	89

LIST OF FIGURES

Figure

2.1	<i>A convolutional neural network architecture.</i> Feature extraction consists of two operations, convolution and pooling. Filters rotate across the input, breaking down different local regions into feature maps to create the convolution layer. The feature maps are downsized via pooling, making the data easier for the final layers to process. During the classification stage, the fully connected layers detect patterns in the image data, now represented as feature maps.	8
2.2	<i>Examples of facial images from the MORPH data set.</i> The key feature that makes the data easier for the machine is the standard positioning and lighting of each subject.	12
2.3	<i>Sample sizes and class proportions for the MORPH data set (majority class in parentheses).</i> The upper left panel shows the baseline sample sizes, with the training data initially having an equal ratio of white to non-white subjects in the sample. The three treatments made to the training data include (1) relative size of the majority (2) which class composes the majority and (3) the sampling method used to obtain the imbalance.	13
2.4	<i>Sample sizes and class proportions for the Fair Face data (majority class in parentheses).</i> The upper left panel shows the baseline sample sizes, with training data having an equal ratio of white to non-white subjects. The three treatments made to the training data include (1) relative size of the majority (2) which class composes the majority and (3) the sampling method used to obtain the imbalance.	14
2.5	<i>Examples of face images from the FairFace data.</i> Constructed to help researchers combat algorithmic bias, this data has more variation, both in the diversity of its subjects and in the images themselves. Subjects might be wearing hats, facing away from the camera, or only partially lit. This data is used to test how well a CNN can be perform on external data, without any additional fine-tuning to its hyper-parameters.	15

2.6	<i>Training metrics for the MORPH baseline classifier.</i> The model shows high accuracy on both training and validation data, presenting the experiment with an opportunity to see how robust these results are to experimental treatments.	17
2.7	<i>Confusion matrices for the MORPH data, random under-sampling, 90% majorities.</i> At extreme levels of class imbalance, accuracy declines for the minority, even in a previously high-performing classifier. F_1 score (majority class W	18
2.8	<i>Confusion matrices for the MORPH data, random over-sampling, 90% majorities.</i> The decline in performance seems specific to under-sampling. When over-sampling is used to achieve imbalance, accuracy remains high. Both models trained on white and non-white majority classes achieve an F1 score of .96	18
2.9	FairFace Baseline	19
2.10	The Fair Face baseline classifier, with the <i>training data</i> composed of equal amounts from each class, makes a poor predictor when classifying the unseen <i>test data</i> . This provides the experiment with an opportunity to see if sampling techniques can remedy poor classification performance.	20
2.11	Random undersampling using the Fair Face data does not correct alleviate an already poor performing baseline classifier. Any improvement in the race classifier in (b) and (c) is specific to the class who's become the majority, due to the undersampled alternative class. . .	21
3.1	<i>A sample tweet showing a positive polling place experience:</i> "just voted today , well organized by @ clarkcountynv . vote early or vote on election day ! its our duty as american ci & just voted today , well organized by @ clarkcountynv . vote early or vote on election day ! its our duty as american citizens . # leadright2016"	32
3.2	<i>Class proportions across the training, validation, and test sets.</i> The original data is subset, with the ratio of non-incidents (0) to incidents (1) preserved. The bulk of the data (n=7950) is used for training, while the validation set (n=500) is used to check for overfitting and the test set (n=500) is used to gauge the final model's predictive power.	36
3.3	<i>Schematic of the image layers from the text-image model.</i> Convolutional filters create feature maps while pooling layers reduce the size of the data. This visual is specific to the image side of the architecture. The text side is parallel, with one-dimensional matrix operations instead of the two-dimensions required for images. . . .	38
3.4	<i>The text-image model's parallel architecture.</i> Content on both sides is represented in matrix form. A combination of transfer learning (from the VGG16 image classifier and GloVe word embeddings), convolutions, and pooling layers extract features from a tweet's content. Once their features have been joined, dense layers learn patterns from both text and image features, before making a classification decision.	46

3.5	<i>Training and validation metrics for the text-only model.</i> The most striking element about the text-only model is not its high training accuracy, but rather, the disparity between training and validation metrics. The model suffers from <i>overfitting</i> . The remedy for this is to code more training data so the classes become more distinct, their features more identifiable by the algorithm.	47
3.6	<i>Confusion matrices for the text-only model.</i> There is a significant performance decline when the text-only algorithm attempts to classify previously unseen data. The model learns to perfectly classify tweets in the training data (a) but only manages to detect sixty percent of election incidents in the test set (b).	48
3.7	<i>Training and validation metrics for the text-image model.</i> The text-image classifier does not perform as well as the text-only classifier on the 2016 Election Twitter data. Although multimodal content classification remains an important goal for the analysis of social media data, not all classification schemes are going to benefit from multimodal input.	49
3.8	<i>Confusion matrices for the text-image model.</i> The poor performance on the training data indicates that a multimodal classifier would require a larger number of <i>incidents</i> in the training data to properly distinguish between features.	50
4.1	THIS is how you fight for justice. #Ferguson #PoliceBrutality #BlackLivesMatter http://t.co/FPIsVeRExR @JNicoleBK	62
4.2	<i>Sample sizes and class proportions across the training, validation, and test data.</i>	65
4.3	<i>Tweets supporting Black Lives Matter</i>	67
4.4	<i>Tweets opposing Black Lives Matter</i>	68
4.5	<i>Schematic of the image layers from the text-image model.</i> Convolutional filters create feature maps while pooling layers reduce the size of the data. This visual is specific to the image side of the architecture. The text side is parallel, with one-dimensional matrix operations instead of the two-dimensions required for images. . . .	69
4.6	<i>The text-image model's parallel architecture.</i> Content on both sides is represented in matrix form. A combination of transfer learning (from the VGG16 image classifier and GloVe word embeddings), convolutions, and pooling layers extract features from a tweet's content. Once their features have been joined, dense layers learn patterns from both text and image features, before making a classification decision. . . .	70
4.7	<i>Training metrics for the text-only model.</i> The model's training and validation metrics immediately diverge. Even though the model shows high training accuracy, predictive performance on the unseen validation data steadily declines.	74

4.8	<i>Confusion matrices for the text only model.</i> The text only model suffers from overfitting, with a noticeable decrease in performance from training to validation. Note especially the upper left panel in (b). The text only model has trouble identifying attitudes Opposed to BLM in the validation data.	75
4.9	<i>Training and validation metrics for the text-image model.</i> Early results show parallel accuracy metrics for the training and validation data. The metrics eventually diverges, however, as the model's parameters overfit to the training data.	76
4.10	<i>Confusion matrices for the text-image model.</i> The model demonstrates high accuracy in identifying supporting tweets even on the validation set. The model does a decent job of identifying opposition tweets on the training data, but less so on validation set. Note that the text-image model does a better job of predicting opposition tweets in the validation set than the text-only model. This suggests that image features supplements information for minority classes. .	77
4.11	<i>Confusion matrix for the text-image model on the 2020 test data.</i> Data was collected in May and June 2020. Results indicate that, like the validation data, the model struggles to classify opposing tweets.	78

ABSTRACT

Social media data enables political scientists to observe phenomena that have been otherwise difficult to capture. The scale and structure of such data is problematic, however, as sorting social media posts by hand is a prohibitively costly endeavor. For instance, there are over 500 million tweets posted per day, consisting of text, image, gif, and video content. This has created a technology gap between what social scientists want to do, conceptually, and what they can do, computationally. This study develops text- and multimodal (text and image) classification technology. Such methods are used to investigate questions in algorithmic bias, election experiences, and Black Lives Matter protest activity. Multiple machine learning algorithms – called convolutional neural networks or *deep learning* models – were developed. These models were trained on facial images and tweet text. Results indicate that deep learning achieves high accuracy on training data; performance declines when the machine attempts to predict the previously unseen validation set. These algorithms can lack predictive power. Deep learning shows promise for automated content analysis, but more work must be done to curate theoretically motivated training data. Social scientists should focus on features in the data that best differentiate categories of interest. This study contributes to larger trends in computational social science that seek to apply machine learning methods to problems in political science. Even the most advanced methodology, however, must be wrapped in strong theory and substantively interesting questions.

CHAPTER I

Introduction: Of Politics and Computers

This project explores subjects at the intersection of politics and computers. The goal of this work is to use advanced computing techniques to capture concepts relevant to the academic study of *politics*. For various reasons, these concepts are difficult to quantify and difficult to study. The following chapters make innovative use of *machine learning* methods, text data, and image data, to quantify political phenomena like elections and protests. As such, this dissertation contributes to two sub-fields in political science: political methodology and political communication.

Within the field of political methodology, a growing body of research advances automated content analysis. This is spurred by the availability of politically significant content found online. Whether from social media or news sites, parliamentary or court transcripts, this seemingly infinite supply of content has fueled demand for efficient methods to extract insight from messy, unstructured data. This data spans a range of modes, including text, image, audio, and video.

Automated analysis across political communication spans multiple areas of interest. Recent research uses such machine learning methods to characterize protests, measure crowd size at campaign events, capture candidates' emotions, and examine politicians' social media strategies (*Torres, 2018; Zhang and Pan, 2019; Joo and Steinert-Threlkeld, 2018; Boussalis and Coan, 2020; Peng, 2020*). The three studies that compose the body of this work use image and text data – and a class of algorithms called *deep learning* – to explore some element of American politics. Substantively, these chapters explore racial and ethnic politics, elections, and social movements.

Deep learning algorithms are a type of supervised machine learning, where computers learn by example. The advantage of this approach is that researchers control how to operationalize their variables (as opposed to the unsupervised learning approach, where the algorithm's output must be interpreted and mapped to a previously un-defined concept). In supervised machine learning, a coding scheme is designed to

capture the concept of interest; categories are defined, rules for membership in each category are established. Then a subset of the data is sorted according to this coding scheme. Finally, this subset is input into the algorithm for training. The end goal is a piece of code that can replicate human classification, in a fraction of the time and without the necessity of training and re-training research assistants. Findings indicate that deep learning models are more than capable of capturing political phenomena in unstructured, messy data – like image and text content.

The first study examines racial politics in the context of *algorithmic bias*. This occurs when a machine learns prejudice somewhere during the training process. As machine learning applications grow ubiquitous in the real-world, so do the consequences of algorithmic bias; for instance, deep learning models are currently deployed by surveillance systems used in the United States. This study tests under what conditions these types of models learn racial prejudice.

The second study examines election administration. Specifically, it captures *election incidents*: anecdotes of voters having either a positive or negative voting experience. When aggregated, these election incidents illustrate when, and where, voters face difficulty casting a ballot. This study captures election incidents during the 2016 U.S. presidential election.

The third and final study examines *political attitudes*. Specifically, this chapter captures attitudes toward the Black Lives Matter (BLM) movement. Capturing attitudes toward Black Lives Matter provides scholars with deeper insight into public opinion on racialized in the United States. Difficulty arises, however, because public opinion toward BLM – like the movement itself – is too dynamic for scholars to keep pace. This study captures political attitudes toward BLM during the 2014 and 2020 wave of protests.

This dissertation explores the intersection of politics and computers, using machine learning to capture concepts relevant to the study of American politics. The following chapters demonstrate how *deep learning* methods can analyze text and image data, helping political science scholars examine elections and protest activity. These methods show promise for automated analysis in political communication.

CHAPTER II

Algorithmic Bias in Deep Learning: An Inquiry Into Race Recognition

Surveillance systems fitted with deep learning technology are currently deployed by both the United States and Chinese governments. Questions have arisen as to the fairness of these systems, the concern being that governments will use deep learning technology to target ethnic minorities. Researchers currently seek methods to audit surveillance systems for algorithmic bias. This study contributes to this effort, using a sampling experiment to test under what conditions deep learning classifiers exhibit algorithmic bias. The experiment artificially induces class imbalances in training data, repeatedly training a race classification algorithm on different subsets of face images. Results indicate that a highly accurate classifier is not biased via intermediate levels of imbalance, but does exhibit bias when training data features more extreme levels of class imbalance. Additionally, the race classifier performs poorly predicting out of sample data, indicating that "out of the box" machine learning models deployed in real-world settings should constantly be monitored, audited, and s(re)tested.

2.1 Introduction

Thou shalt not make a machine in
the likeness of a human mind.

Frank Herbert, *Dune*

When governments abuse power, it can be difficult to combat the abuse without understanding their tactics and techniques. The Watergate scandal does not make much sense without the word "wiretapping." It is telling that most famous whistleblower of the past fifty years, Edward Snowden, was a *computer security consultant* working for the National Security Agency (NSA). From wiretapping to email surveillance, technology can be a black box large enough to obscure tyranny. Currently, both the United States and Chinese governments deploy machine learning in surveillance systems used by law enforcement (*Mozur, 2019*). Little is understood about the underlying technology, but concerns among journalists and scholars have arisen over whether surveillance systems meant to identify demographic information – like race or ethnic identifiers – can remain impartial.

This paper tests how sensitive machine learning (ML) is to human bias. This study explores *algorithmic bias*: "when seemingly innocuous programming takes on the prejudices either of its creators or the data it is fed" (*Garcia, 2016*). Decision making systems powered by machine learning are pitched to the public as replacing human bias with objective considerations of data. The reality is that machines are more than capable of learning human bias. In the United States, loan eligibility software considers ZIP code data when making decisions, police use facial recognition to decide what populations to monitor – and both applications have delivered discriminatory results, despite the promise of computers' impartiality (*Du et al., 2020*). As machine learning becomes more ubiquitous in the real-world, the presence of algorithmic bias grows more consequential.

The fear is that policy diffusion will normalize the use of race recognition software by authoritarian governments targeting specific minority groups. Jonathan Frankle, an artificial intelligence researcher at the Massachusetts Institute of Technology (MIT) warns race classification poses an existential threat to democracy. Referencing the Chinese government's use of race recognition to identify – and intern – ethnic Muslims, Frankle notes:

Once a country adopts a model in this heavy authoritarian mode, it's using data to enforce thought and rules in a much more deep-seated fashion than

might have been achievable 70 years ago...To that extent, this is an urgent crisis we are slowly sleepwalking our way into (*Mozur*, 2019).

At the intersection of society and technology, political science is uniquely positioned to investigate the social perils of machine learning.

While machine learning researchers have started exploring algorithm bias from an ethical, abstract perspective, the larger social implications of this technology requires input from computational social science. Current research has failed to empirically capture algorithmic bias, or even consider its real-world implications: "[studies] have tried to draw attention to this issue from a conceptual standpoint without much empirical effort to shed light on its behavioural, organisational, and social impacts" (*Kordzadeh and Ghasemaghaei*, 2022, p. 404). This paper takes a step toward that end, introducing social scientists to the language of algorithmic bias research and empirically examining the conditions that create racist machines.

While political communication scholars have begun using neural networks – or *deep learning* models – to automate content analysis, it is important that the political implications of these algorithms also drive discussion (*Garcia*, 2016; *Torres and Cantú*, 2022). The proliferation of this knowledge not only advances applications in academic political science, but also arm researchers with the language and technical detail necessary to investigate potential bias.

Questions remain as to *where* in the process ML practitioners should audit algorithms for bias (*Islam et al.*, 2022). This study uses sampling experiments to show race recognition classifiers are sensitive to imbalances in training data. Identically configured models were trained – their tuning parameters unchanged – with manipulations made to the class composition of the training data. Treatment variables include: the *level* class imbalance; the sampling *method* used to achieve the imbalance; and which *racial group* was chose as the minority class. The sample is manipulated via the random over- and under-sampling of white and non-white observations – this results in multiple training sets with varying degrees of bias. The paper finds that imbalanced training samples negatively impact a classifier’s ability to identify underrepresented classes, even if overall performance appears strong. These results hold across two different data sets and multiple manipulations to training data.

The rest of the paper is organized as follows. Section 2 details the fairness problem in ML and the sampling methods researchers use to combat bias. Section 3 reviews how deep learning is used for race-feature classification. Section 4 details image processing and experiment design. Section 5 reviews the results and discusses avenues for future research.

2.2 Fairness Problems and Sampling Solutions

This study explores how deep learning methods can be manipulated in a way that has real-world, political consequences. Fairness problems in algorithmic bias research fall into one of two categories – prediction outcome discrimination and prediction quality disparity. *Prediction outcome discrimination* refers to the exclusion from resources because of membership in certain groups. For instance, hiring systems have been shown to discriminate against women because training data featured mostly men. *Prediction quality disparity* occurs when a classifier performs poorly at identifying specific groups, resulting in potentially harmful misclassification error – e.g., surveillance systems misidentifying someone’s race (Du et al., 2020).

Algorithmic bias in facial recognition systems will suffer from prediction quality disparity because of the *under-representation problem*:

Data may be less informative or less reliably collected for certain parts of the population...If the model cannot simultaneously fit all populations optimally, it will fit the majority group. Although this may maximize overall prediction accuracy, it might come at the expense of the under-represented populations and leads to poor performance for those groups (Du et al., 2020, 2).

To combat bias, machine learning practitioners have experimented with strategies at different stages of the training process. Addressing bias during data pre-processing – prior to the actual training of the machine-learning algorithm – has been shown to ameliorate bias at higher rates than post-processing (Islam et al., 2022). Focus has turned to producing more balanced data sets. Researchers correct imbalances either by down-sampling from the majority class (random under-sampling), or repeatedly re-sampling from the minority class (random over-sampling) (Yu et al., 2020).

Random over-sampling, or *ROS*, artificially increases the size of the underrepresented class by duplicating random observations in the sample. When the subsample is taken from the original data set, the random over sampling draws from the minority class with replacement (the majority class is sampled without replacement). The benefit of this approach is that no information is lost prior to training; however, this method runs the risk of the model overfitting because it sees observations multiple times.

The second sampling method, called random under sampling, or *RUS*, decreases the size of the majority class to equal the size of the minority class. The loss of

information from discarding part of the training data results in poorer accuracy performance (*Islam et al.*, 2022). Indeed, when building fair models, researchers often face a trade-off between overall accuracy and class-specific accuracy (debates over the definition of *fair* and what metrics to use to capture the concept are left for other studies in this field).

2.2.1 Machine Learning in Political Communication

The availability of politically rich, digital content has popularized automated analysis across political communication. Machine learning applications span multiple modes and areas of interest. Recent research uses these methods to characterize protests, measure crowd size at campaign events, capture candidates' emotions, and examine politicians' social media strategies (*Torres*, 2018; *Zhang and Pan*, 2019; *Joo and Steinert-Threlkeld*, 2018; *Boussalis and Coan*, 2020; *Peng*, 2020). The deep learning models used throughout the rest of this paper are trained to classify facial images into race-categories. This study explores how these methods can be manipulated in a way that has real-world, political consequences.

2.3 Deep Learning for Race Classification

This section overviews the deep learning models being tested for algorithmic bias in race classification tasks. Deep learning models, also called *convolutional neural networks* or CNN's, are layered structures for processing and classifying high-dimensional data (*Hastie et al.*, 2009). CNN's are considered *feed-forward networks* because data is passed forward from the input layer to the middle and output layers. An example of this process is shown in Figure 2.1. Training a CNN requires the use of weights that differentiate between relevant features in the input data; weights and input data are combined via matrix multiplication before being fed forward to the next layer. The network performs different operations on the data to capture important signals, reduce dimensionality, and find relationships in the data.

There are two phases for a CNN to process content: feature extraction and classification. During feature extraction, *convolutional* layers use filters that rotate across the data, searching for specific patterns and breaking down each individual input into *feature maps*. *Pooling* reduces the dimensionality of the data by either taking the average or the maximum among clusters of data points, reducing noise in favor of the strongest signals.

Between the convolutional and pooling operations, a lot of effort is expended

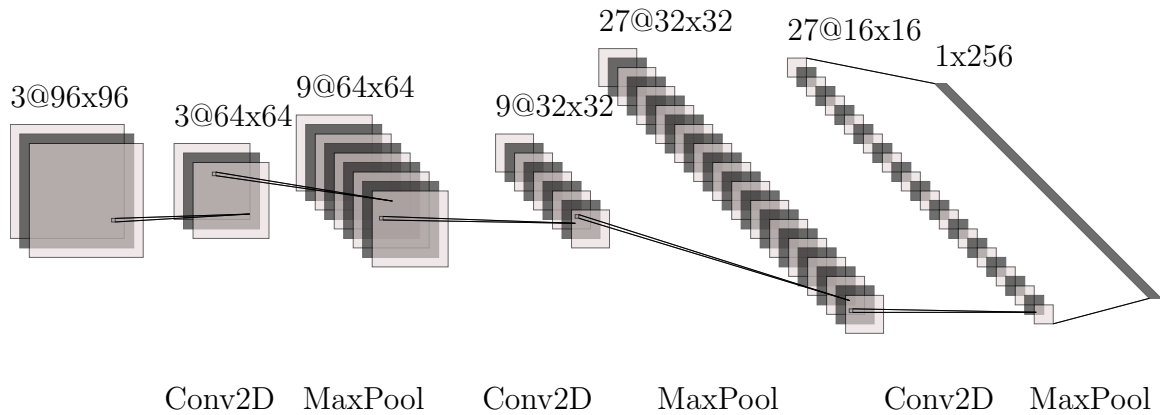


Figure 2.1: A convolutional neural network architecture. Feature extraction consists of two operations, convolution and pooling. Filters rotate across the input, breaking down different local regions into feature maps to create the convolution layer. The feature maps are downsized via pooling, making the data easier for the final layers to process. During the classification stage, the fully connected layers detect patterns in the image data, now represented as feature maps.

reducing images to their most essential features, making data easier to process in the final layers of the network. After feature extraction, the final layers of a CNN are concerned with classification. Feature maps are fed into the *dense* layers – all of the outputs from one layer connect, as inputs, to all functions in the subsequent layer. This enables the network to detect non-linear relationships like image patterns that signal an image’s membership in one class versus another.

2.3.1 Transfer Learning

Spurred by the costliness of training complex CNN’s, machine learning research has explored transfer learning – a technique where a model’s parameters, trained on a *source task*, is applied to a new *target task*. Transfer learning defies the assumption that training and target data must come from the same domain; often, researchers have a target task in one area of interest but use a model pre-trained on data from a separate, more general domain (*Pan and Yang, 2010*). This process works well with image data because images share a feature space. The lower levels of a neural network learn basic patterns, like edges and shapes, that are not exclusive to any particular domain, but rather, common to all pictures. Transfer learning seeks to improve performance by using the parameters of a pre-trained model to extract features from the target data. More formally:

Definition 1. Transfer Learning. Given a source domain D_S and learning task

T_S , a target domain D_T and learning task T_T , transfer learning aims to help improve the learning of target predictive function $f_T(\cdot)$ in D_T using the knowledge in D_S and T_S , where $D_S \neq D_T$, or $T_S \neq T_T$.

Let domain D be defined by a *feature space*, X , and a marginal probability distribution, $P(X)$. For a domain of interest, $D = \{X, P(X)\}$, a task, T , has two parts: a set of labels Y and a predictive function $f(\cdot)$. A task simply refers to a function that predicts labels: $T = \{Y, f(\cdot)\}$. Much like research assistants, the predictive function learns from examples. This training data consists of pre-labeled data $\{x_i, y_i\}$ where $x_i \in X$ and $y_i \in Y$. The ultimate goal is to estimate the parameters of $f(\cdot)$ that can predict the label $f(x)$, given a new instance x .

Assume there are only two domains of interest: a source domain D_S and a target domain D_T . Data sampled from the source domain data is denoted $D_S = \{(x_{S_1}, y_{S_1}), \dots, (x_{S_n}, y_{S_n})\}$ where $x_{S_i} \in X_S$ is a data point and $y_{S_i} \in Y_S$ is the corresponding label. Equivalently, the target domain data is denoted $D_T = \{(x_{T_1}, y_{T_1}), \dots, (x_{T_n}, y_{T_n})\}$ where $x_{T_i} \in X_T$ is the data point and $y_{T_i} \in Y_T$ is the corresponding label. Transfer learning is the process of applying knowledge gained from D_S to a task in D_T . This study uses parameters frozen from the ResNet50V2 model (He et al., 2016).

Model Architecture On top of the the frozen parameters of ResNet50V2 model sit the trainable weights of the adaptation layers. This includes 18 layers whose operations include two-dimensional convolutions (between one and six filters per layer), max pooling, dense (three layers each with thirty-six, eighteen, and nine nodes respectively) and the final output (decision) layer. The ResNet model performs feature extraction. The adaptation layers consist of 224,033 trainable weights that update and learn patterns in the features. For further details on image processing and model architecture, please see the code overview in Appendix I of this chapter.

Parameter Tuning The overall number of layers, the size of the convolution filters, the type of activation function, and the type of pooling are all *tuning-parameters* – adjustable settings defined by the user that impact both the speed of training and the accuracy of the model. An enormous amount of time and energy is spent testing different combinations of tuning-parameters to find what configuration provides the best performance. The difficulty in parameter tuning is that there are no hard and fast rules about how to determine the optimal configuration; no two networks are built the same, as configurations are specific to the problem statement and classification

scheme. There are certain heuristics that guide experimentation (for instance, it is generally best to increase depth with the complexity of the image) but this requires trial and error.

Parameter tuning is a balance between potential improvements and costs to model performance. Increasing the number of filters helps improve accuracy, but this requires more training data. Increasing the size of the learning rate helps optimization converge faster, but runs the risk of unstable training and sub-optimal parameters. A learning rate that is too small will result in a long training process that gets stuck at a local minimum. Too many filters in a model risk *overfitting* – when a model adjusts parameters to accurately classify training data without detecting the relevant patterns that let it classify unseen data. This is why it is standard practice to set aside a subset of the pre-labeled data for the purposes of validation. This *validation data* is not used as input for training, but rather, to test the model’s predictive power at the end of each epoch.

Backpropagation During training, optimal weights are found via a process called *backpropagation*. This process occurs in three stages – the forward pass, the backward pass, and the gradient update. During the forward pass, the linear combination of weights and input data is computed, then fed forward layer-to-layer until the model outputs the predicted label. During the backward pass, the distance between the predicted and true labels is calculated using the *loss function*. The derivative of the loss function with respect to each weight, the *gradient*, is computed layer-by-layer starting from the output layer and going backward. The third step in backpropagation uses the calculated gradient to adjust the weights toward the steepest decrease of the loss function. This process of iteratively computing the gradient of the loss function with respect to each weight and then adjusting weights to minimize loss, called *gradient descent*, efficiently decreases error at each training step.

Benchmarks for Training Training performance is measured using three benchmarks: accuracy, cross-entropy loss, and F_1 score. Accuracy, defined formally in Equation 2.1 below, captures the percentage of images the model correctly classifies. We define y_i as the true value for data point i and \hat{y}_i as the model’s predicted label. The higher the accuracy, the better the model.

$$\text{accuracy}(y, \hat{y}) = \frac{1}{n_s} \sum_{i=0}^{n_s-1} \mathbf{1}(\hat{y}_i = y_i) \tag{2.1}$$

Categorical cross-entropy loss, defined in Equation 2.2 below, is calculated as a sum of separate loss for each class label per observation. As probability p diverges from the true label, cross-entropy increases – loss closer to 1 indicates poor performance, while a perfectly performing model would have a loss of 0.

$$\text{loss}(y, p) = - \sum_{c=1}^M y_{i,c} \log p_{i,c} \quad (2.2)$$

In the above equation, we define M as the number of classes, p as the predicted probability, and $y_{i,c}$ is a binary indicator (0 or 1) that indicates if c is the correct class. The algorithm uses the loss function to capture prediction error. Gradient descent then updates weights to minimize loss, thereby reducing error before attempting to classify another batch of data. Training neural networks is thus framed as an optimization problem: update weights to reduce error until you arrive at a global minimum.

$$F_1 = \frac{2}{\text{recall}^{-1} + \text{precision}^{-1}} = 2 \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}} = \frac{2\text{tp}}{2\text{tp} + \text{fp} + \text{fn}} \quad (2.3)$$

Equation 2.3 above shows how F_1 is computed as the harmonic mean of precision and recall. Precision, conceptually, is the positive prediction rate and recall is a measure of sensitivity. The above equation refers to true positives ("tp"), false positives ("fp"), and false negatives ("fn"). Combining them as F_1 gives a single measure of the model's ability to predict unseen data (the statistic is between 0 and 1, the higher the score the better overall performance).

2.4 Image Processing and Experiment Design

To understand how a CNN processes visual data, think of an image as a collection of pixel values. Images are represented by their pixel values, from 1 to 256, along three color channels – red, green, and blue (RGB). These values are stored in a three-dimensional matrix – one dimension per color channel. Data is normalized by dividing all values by the range (255). This ensures that all of the input data exists on the [0,1] scale which enables the network to converge faster during training.

During training, the CNN performs a series of matrix transformations on the data, so there cannot be any variation in dimensionality of the inputs. The training algorithm expects all images to have a square, uniform size, so all images are resized to 96 x 96. For data augmentation, pre-processing images can include transformations

to provide the model more variation from its training input. Training images can be randomly flipped, zoomed-in, or tilted along different axes; this increases variation in the features that the machine sees during learning.

MORPH While MORPH was originally used for research into aging, political communication research has used this data, with transfer learning, to build a race classifier for profile pictures on Twitter (*Wang et al.*, 2016). This data set contains 55,000 unique images of over 13,000 individuals from 2003 to 2007 and corresponding meta-data on gender, age, and ancestry (*Ricanek and Tesafaye*, 2006). Examples of the MORPH images are shown in Figure 2.2. The data is longitudinal, so the data features multiple images of the same subject over time (the average number of images per subject is four).



Figure 2.2: *Examples of facial images from the MORPH data set.* The key feature that makes the data easier for the machine is the standard positioning and lighting of each subject.

For this study, the original data set of 55,134 photos were split into a test set ($n = 11,026$) a validation set ($n = 11,026$) and a training set ($n = 33,082$). The *Ancestry* categorical variable originally featured designations for white, African-American, Hispanic, Asian, and Native-American. For the simplicity of the study, these categories were collapsed and the race label was a simple *white* or *non-white* binary variable. The data sets were then balanced so that they have equal proportion of *W* and *NW* labels. For the training set, this resulted in a baseline sample size of 12,622 (each class had $n = 6,311$ images). The baseline validation set used in the confusion matrices below have a sample size of 4,296 (each class had $n = 2,148$ images). Over the course of the experiment, the sample size of the training set was manipulated according to the treatments outlined in Figure 2.3. Validation data was left untouched.

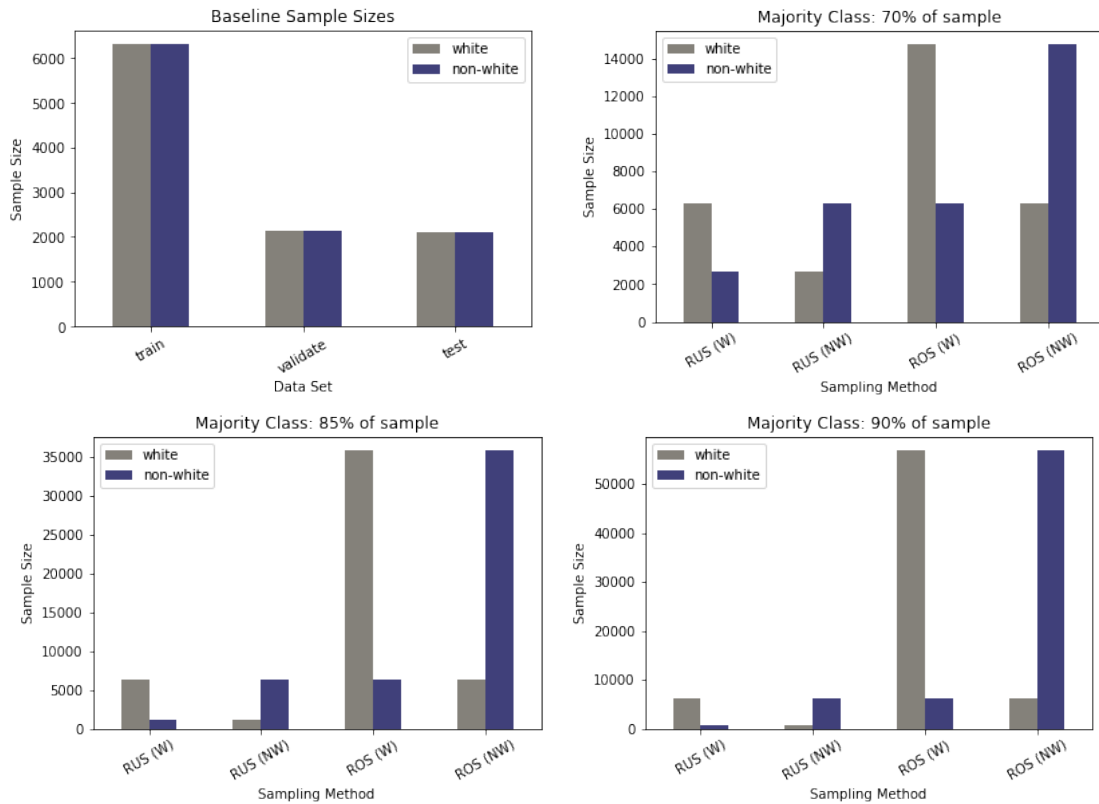


Figure 2.3: *Sample sizes and class proportions for the MORPH data set (majority class in parentheses).* The upper left panel shows the baseline sample sizes, with the training data initially having an equal ratio of white to non-white subjects in the sample. The three treatments made to the training data include (1) relative size of the majority (2) which class composes the majority and (3) the sampling method used to obtain the imbalance.

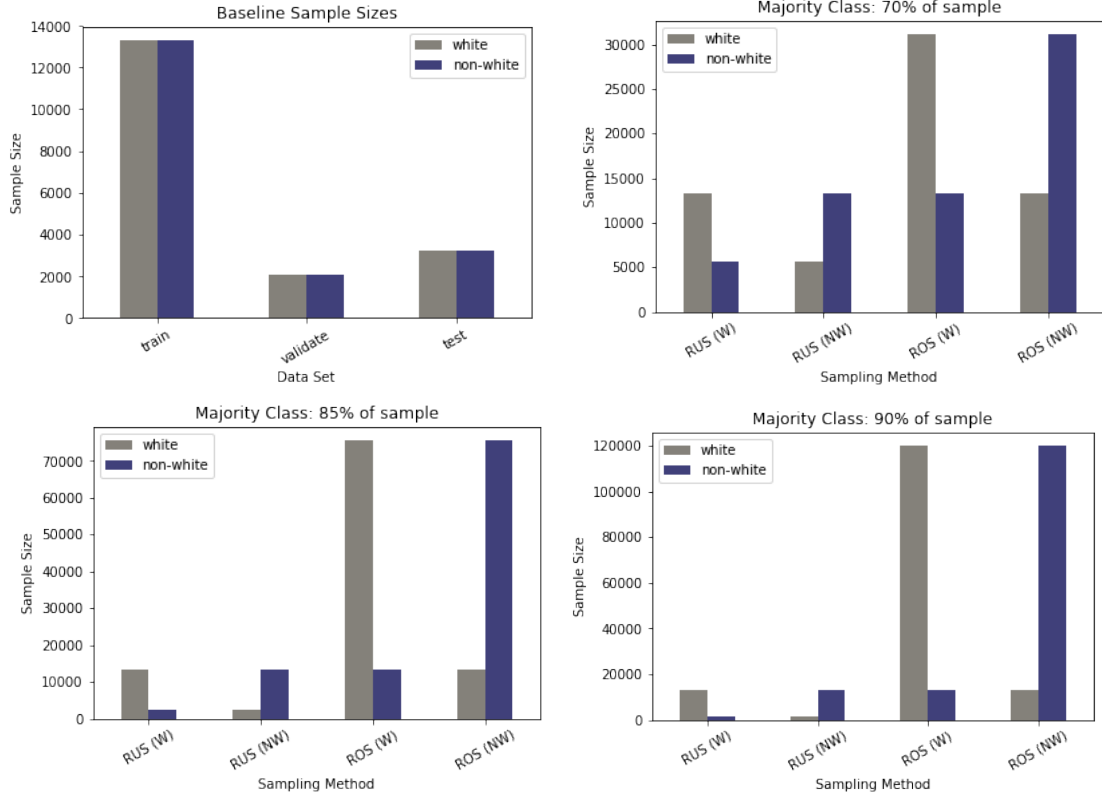


Figure 2.4: *Sample sizes and class proportions for the Fair Face data (majority class in parentheses).* The upper left panel shows the baseline sample sizes, with training data having an equal ratio of white to non-white subjects. The three treatments made to the training data include (1) relative size of the majority (2) which class composes the majority and (3) the sampling method used to obtain the imbalance.

2.4.1 Experiment Design

The sampling experiment used three treatments. The first treatment was the *sampling method* used to manipulate the training data; random under- and over-sampling induced artificial imbalances in the data’s class distribution. The second treatment was which label became the *majority class*; data was sampled to feature a clear majority, either the white (W) or non-white (NW) race-category. The third treatment was the *degree of class imbalance*. The training data featured either low, moderate, or extreme levels of imbalance – this corresponded to the majority class composing 70%, 85%, and 90% of the overall sample, respectively. The “baseline” model had an even split across both classes. The classifier was trained on the resulting imbalanced training set and then tested on the balanced validation set.

External validation using FairFace The FairFace data was built specifically to mitigate bias in image data sets (Karkkainen and Joo, 2021). About 100,000 observations were collected from YFCC-100M Flickr data set, which includes labels for race, gender, and age (examples from the data set are shown in Figure 2.5). The FairFace data over-sampled racial minorities to provide deep learning models with additional examples of a traditionally underrepresented groups (Karkkainen and Joo, 2021). Indeed, the race label consisted of seven categories: White, Black, Indian, East Asian, Southeast Asian, Middle Easter, and Latino. For the simplicity of the study, these categories were collapsed and the race label was a simple *White* or *non-White* binary variable.



Figure 2.5: *Examples of face images from the FairFace data.* Constructed to help researchers combat algorithmic bias, this data has more variation, both in the diversity of its subjects and in the images themselves. Subjects might be wearing hats, facing away from the camera, or only partially lit. This data is used to test how well a CNN can be perform on external data, without any additional fine-tuning to its hyper-parameters.

Over the course of the study, the original data set of 97,698 photos was split into a test set ($n = 17,348$) a validation set ($n = 10,954$) and a training set ($n = 69,396$). The data sets were then balanced so that they have equal proportions W and NW labels. For the training set, this resulted in a baseline sample size of 26,632 (each class had $n = 13,316$ images). The baseline validation set had a sample size of 4,170 (each class had $n = 2,085$ images). The baseline test set had a sample size of 6,422 (each class had $n = 3,211$ images). Over the course the experiments, the sample size of the training set was manipulated according to the treatments in Figure 2.4.

2.5 Results and Discussion

The race-classification algorithm showed resistance to bias at low and moderate levels of class imbalance. Once the algorithm was exposed to extreme levels of class imbalance, however, it showed a predisposition to predict images as belonging to the majority class, regardless of the true label. This finding holds regardless of which race-category was the majority class. The classifier initially performed well on the MORPH data set, as shown in Figure 2.6. It eventually became biased when the majority class composes ninety percent of the training sample, achieved via random under-sampling of the minority class. The bias disappears if the imbalance is created via random over-sampling. These findings indicate that deep learning models are resistant – but not immune – to algorithmic bias induced via class imbalances in the training data. As such, when auditing deep learning algorithms in the real-world, researchers should also audit the training data the algorithms receive as input.

MORPH The baseline model, trained on balanced data, accurately classifies both training and validation data (F_1 score: .96). The baseline gave the experiment an opportunity to see if a high-performing race classifier becomes biased when sampling techniques manipulate the training data. The model continued to perform well on imbalanced data sets, even with majority classes composing seventy or eight-five percent of the overall training set. On these data sets, the classifier hits about ninety-five percent accuracy across both W and NW classes, regardless of which class is in the majority, or the sampling method used to achieve the imbalance.

Predictive performance declines, however, when the model is trained on a highly imbalanced data set, when either white or non-white subjects compose ninety percent of the overall sample. As shown in Figure 2.5, extreme levels of imbalance result in class-specific declines in accuracy; that is, the training sample is so biased toward one class, the model’s parameters default predictions toward that class. For instance, the confusion matrix in Figure 2.7(a) features a training sample of ninety-percent non-white subjects (this imbalance is achieved via the random under-sampling of white subjects).

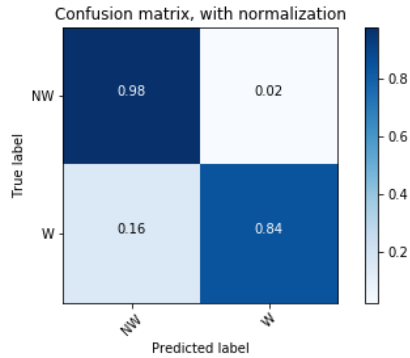
The top-left and bottom-right panels of the confusion matrix indicate the proportion of *positive* identifications for the respective classes. The bottom left panel indicates the proportion of White subjects that the model misidentified as Non-White. Its opposite, the top right panel, indicates the the proportion of Non-White subjects the model misidentified as White.



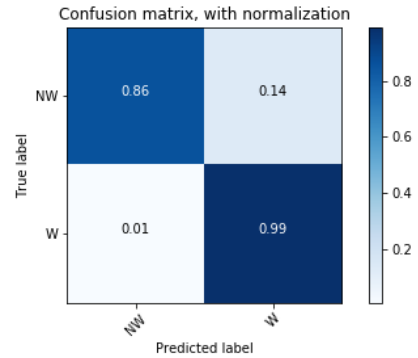
Figure 2.6: *Training metrics for the MORPH baseline classifier.* The model shows high accuracy on both training and validation data, presenting the experiment with an opportunity to see how robust these results are to experimental treatments.

When ninety percent of the training data features NW labels, as in Figure 2.7(a), the model is more likely to misidentify white subjects as being non-white. These results are reversed when the majority class is reversed, as in Figure 2.7(b). Even previously high-performing classifiers can become biased if their training data is manipulated enough. If either of these classifiers were used by law enforcement for suspect identification, for instance, the seemingly objective computer would be biased toward one race category or another.

FairFace The baseline FairFace classifier performs well on training data, but poorly on validation sets, as seen in Figure 2.9 (F_1 score of .68). This was expected, as no additional fine-tuning was done after the adaptation layers of the algorithm were trained on the MORPH data. The FairFace data was brought in as a robustness check for additional validation. The point of the experiment was to see if the original

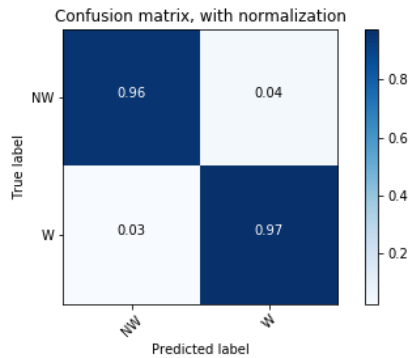


(a) RUS // training set 90% NW

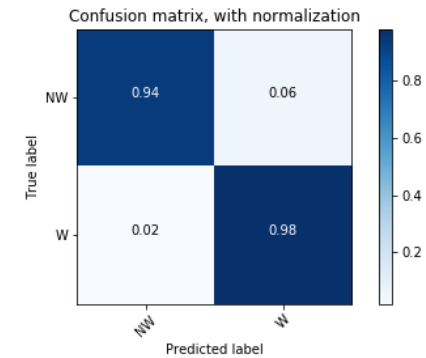


(b) RUS // training set 90% W

Figure 2.7: *Confusion matrices for the MORPH data, random under-sampling, 90% majorities.* At extreme levels of class imbalance, accuracy declines for the minority, even in a previously high-performing classifier. F_1 score (majority class W) : .92. F_1 score (majority class, NW) : .91.



(a) ROS // training sample: 90% NW



(b) ROS // training sample: 90% W

Figure 2.8: *Confusion matrices for the MORPH data, random over-sampling, 90% majorities.* The decline in performance seems specific to under-sampling. When over-sampling is used to achieve imbalance, accuracy remains high. Both models trained on white and non-white majority classes achieve an F_1 score of .96

model could be re-trained on out of sample data, without any additional changes to the tuning parameters. The answer is no, not to any useful extent.

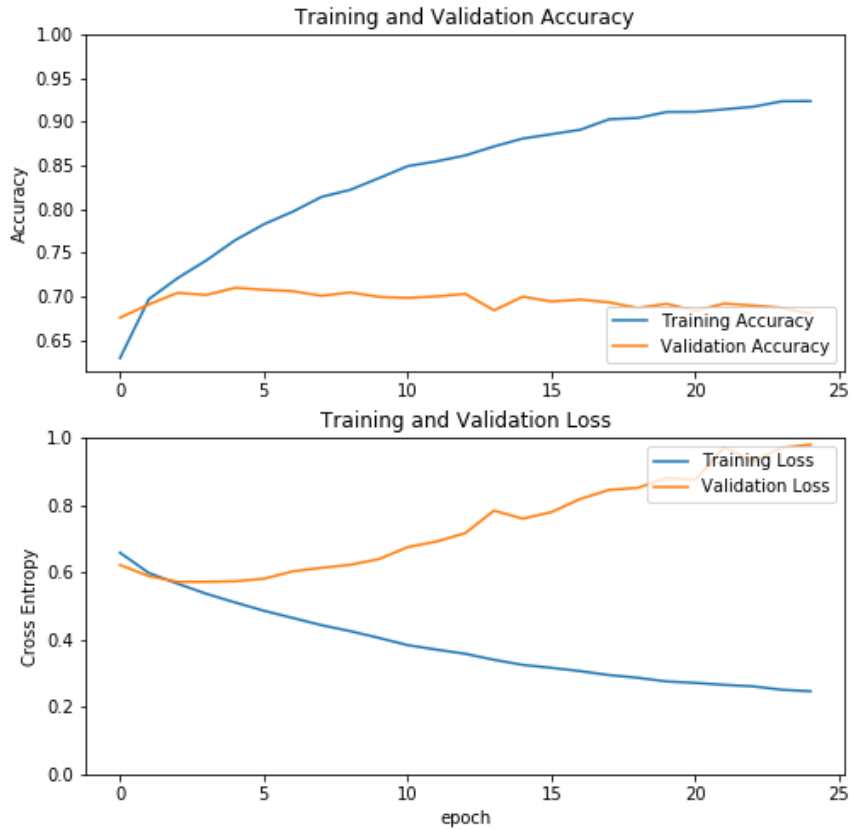


Figure 2.9: FairFace Baseline

The baseline FairFace classifier performed equally poorly on both class labels, enabling the study to explore if sampling alone can improve performance. Looking at the confusion matrix in Figure 2.10, the false positive rate (lower left and upper right panels) are similar across both the NW and W labels. The FairFace baseline classifier falsely predicts label W on 33% of the NW data; it predicts label NW on 31% of the W data. This provides the experiment with an opportunity to see if sampling techniques alone can remedy poor classification performance.

The FairFace results showed that a poorly performing classifier can be improved via sampling techniques, but at the expense of the minority class. Looking at Figure 2.11(a), this model's training data *under-sampled* the images from the White category such that non-Whites composed 70% of the training set. The results are an improved

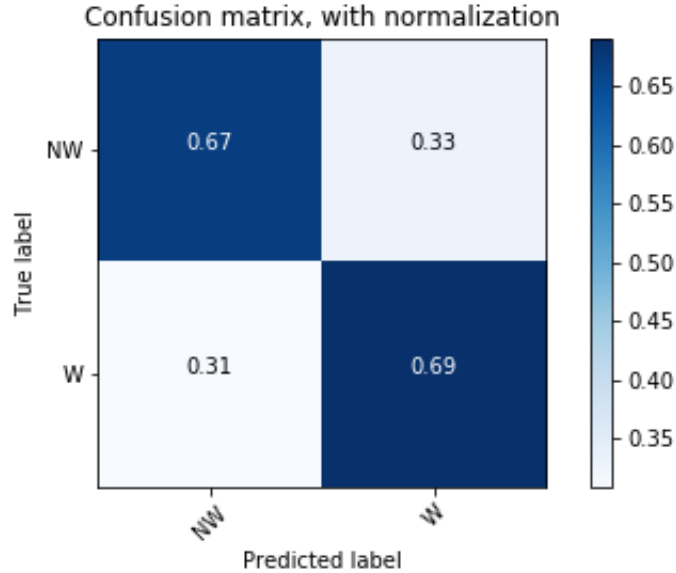


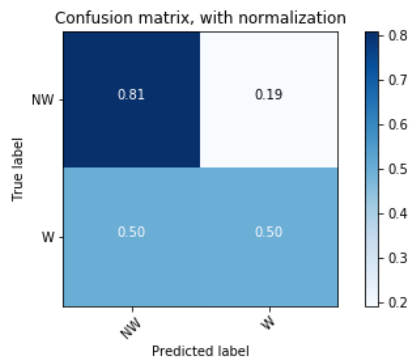
Figure 2.10: The Fair Face baseline classifier, with the *training data* composed of equal amounts from each class, makes a poor predictor when classifying the unseen *test data*. This provides the experiment with an opportunity to see if sampling techniques can remedy poor classification performance.

performance (compared to baseline) when the true label is *NW*, but decreases the classifier’s ability to accurately predict label *W*. The model suffers from prediction quality disparity because of the under-representation problem.

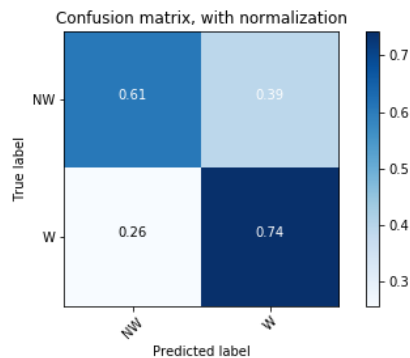
These results hold when the majority and minority classes are switched. When the non-White images are under-sampled such that the training data is composed of 70% White images, we find only a slight difference in performance, compared to the baseline model. As seen in Figure 2.11(b), the classifier’s accuracy rate (.74) is higher than baseline’s (.69) when classifying White images, but down (.61 from .67) when attempting to classing non-White images. Without parameter tuning, a poorly performing classifier will be more susceptible to algorithm bias induced via class imbalance.

2.5.1 Discussion

This study examined the relationship between algorithmic bias, sampling, and deep learning. The goal was to see under what conditions a previously unbiased race classification model would inherit bias. Two different sampling techniques were used to manipulate class balances in training data, with the same model configuration being re-trained on different subsets of data. Results indicate that high-performing



(a) RUS. training set: 70% NW



(b) RUS. training set: 70% W

Figure 2.11: Random undersampling using the Fair Face data does not correct alleviate an already poor performing baseline classifier. Any improvement in the race classifier in (b) and (c) is specific to the class who’s become the majority, due to the undersampled alternative class.

deep learning models are resistant to certain levels of class-imbalance, but eventually become biased when trained on more extreme levels of imbalance.

Given these results, future research into algorithmic bias should focus less on abstract, ethical definitions of "fairness" and instead devote energy on curating balanced data sets. Race recognition models inherit bias from under- or over-represented groups in the training data. Research should focus on auditing both the training data *and* the deep learning algorithms. Just because a machine learning algorithm show high levels of training accuracy, this does not necessarily translate to high predictive performance on previously unseen, real-world data. Further, an algorithm’s accuracy is irrelevant if its application is a violation of human rights.

If police in the United States are already pre-disposed to arresting racial minorities, then the (mis)use of deep learning technology to do so is yet another example in larger patterns of historical oppression. The methods in this paper can help combat algorithmic bias, but combating systemic racial oppression should be the greater concern. Perhaps the most disturbing application of machine learning I have read is the Chinese government’s efforts to identify and intern ethnic Muslims located in the country’s Western region. This sets a dangerous precedent: "A new generation of start-ups catering to Beijing’s authoritarian needs are beginning to set the tone for emerging technologies like artificial intelligence. Similar tools could automate biases based on skin color and ethnicity elsewhere" (*Mozur*, 2019). If deep learning models become yet another black box for governments to hide human rights abuses, then the problem society must confront is not biased algorithms, but rather, authoritarian

governments.

One question that arises when assessing these models: how do we know that these results do not stem from improper tuning parameters, or simply poorly performing models? The motivating question was to explore under what conditions machines learn racism. The experimental framework helped reveal when machine learning errors are not random, but rather, systematically tied to imbalances in the training data. For the most part, the algorithms were resistant to manipulations to the data; if the algorithms trained on low or moderate levels of imbalance were deployed in a real-world setting – like law enforcement efforts to identify the racial-identity of a subject – those models would not be biased toward one category or another. If the algorithms trained on extremely imbalanced data were deployed in any real-world setting, however, they would be biased toward over-identifying the majority class. Perhaps the most important takeaway is that even previously accurate algorithms gradually become biased when training data shifts toward one class over the other.

This study used three experimental conditions to test for the robustness of these results. A key experimental treatment was the proportional size of the majority. This variable was captured as if it were discrete – class imbalance was set to either low, moderate, and high level of imbalance. Future studies should capture a continuous variable. This type of measure would enable the creation of confidence intervals. Bootstrapping could be used to construct confidence intervals around the estimated test score. In doing so, my results can establish that the bias – or lack thereof – exhibited by the model is not specific to just one sub-sample.

A second question that arose during this study: how would this experiment be used to audit real-world software? Researchers would need to purchase a software license from a vendor – unless the study was trying to test multiple surveillance systems, then multiple licenses would need to be purchased. The difficulty in this regard is that, unlike in academia where code and data are shared for the sake of transparency and replication, proprietary software is not be freely available. In many instances, the code and training data represent the product being sold. Any testing of such software would require external data sets and be limited by whatever tuning parameters the software allows.

Finally, future researchers may be interested in exploring whether or not these results hold for specific racial-identities, as opposed to the vague "non-white" class used throughout this paper. To answer this question, this experiment could be replicated almost identically, but with original *Black*, *LatinX*, and *Asian* race-identifiers replacing the non-white category. The difficulty would be all of the potential permutations

such categories would allow in an experimental setting. A comprehensive study would require that the machine learner be tested for bias across all racial pairings; that is, the experiment would have to test whether a machine trained on images of mostly Black subjects would misclassify LatinX as such, and vice versa.

Deep learning models are not immune to imbalances in training data, but they are resistant. While such algorithms are not easily manipulated once their weights have been optimized, getting to that point requires an enormous amount of trial and error. It is not always clear what features are confusing the classifier, or what elements of the sample are most recognizable as belonging to one class or another. The easiest, perhaps lowest level of criticism of deep learning is that such models are a "black box." I argue something is only a black box if you are too scared – or too ignorant – to look inside. The current trend in deep learning is to better understand the middle layers of the network, where mid-level features are represented by partially trained parameters. Social scientists should continue to explore machine learning not only from a methodological standpoint, but also to understand the social implications of such technology,

2.6 Appendix I: Code Overview

```
#!/usr/bin/env python3
# coding: utf-8

# Pseudo-code for Race Recognition Model: Algorithmic Bias
in Deep Learning
DOI: 10.13140/RG.2.2.28514.61126

# For full code, visit:
https://github.com/apineda91/DeepLearn4PolComm
# python libraries required: tensorflow, numpy, sklearn,
    nlt
k, pandas, keras (part of tensorflow), matplotlib, os, sys

# Define functions to balance (and rebalance) classes

def class_balance(dat):
def new_balance(dat, z, samp_method, majority_class):

# Define function to plot_confusion_matrix
def plot_confusion_matrix(cm, classes, normalize=False,
    title
='Confusion matrix', cmap=plt.cm.Blues):

# Specify data you wanna work with: MORPH or FairFace
    # Load training data
    # Specify data locations
    # Specify cross-validation proportions

#### Split data into train/test/validate ####
    # Shuffle data
    # Test data (set aside for final model)
    # Validation data (for parameter tuning)
    # Training data (for training)
```

```

# For FairFace data create race category variable
  # Create the dictionary
  # Add a new column named 'race_cat'
  # map FairFace values to White or Non-White variable

# Data wrangling:
  # Balance datasets as necessary
  # Check datatypes
  # Check counts for race variable (across all subsets)
  # Make sure all images have a valid path and are where
    y
ou think they are
  # Make sure sample sizes are divisible by batch number
  # Specify image size, number of classes, and directory
    p
ath

# Initialize data generators

# Create the image data generator
img_datagen = ImageDataGenerator(
    #featurewise_center=True,
    #featurewise_std_normalization=True) #
    standardizati
on
    rescale = 1./255) # normalization

# Use it to load the different datasets
train_generator = img_datagen.flow_from_dataframe(
    train_dat,
    directory=train_dat_location,
    x_col='full_path',
    y_col='race_category',
    #weight_col=None,
    target_size=(img_width, img_height),
    color_mode='rgb',

```

```

        classes=None,
        class_mode='categorical',
        batch_size=batch_size,
        validate_filenames = False,
        shuffle = False)

val_generator = img_datagen.flow_from_dataframe(
    val_dat,
    directory=val_dat_location,
    x_col='full_path',
    y_col='race_category',
    #weight_col=None,
    target_size=(img_width, img_height),
    color_mode='rgb',
    classes=None,
    class_mode='categorical',
    batch_size=batch_size,
    validate_filenames = False,
    shuffle = False)

test_generator = img_datagen.flow_from_dataframe(
    test_dat,
    directory=test_dat_location,
    x_col='full_path',
    y_col='race_category',
    #weight_col=None,
    target_size=(256, 256),
    color_mode='rgb',
    classes=None,
    class_mode='categorical',
    batch_size=batch_size,
    validate_filenames = False,
    shuffle = False)

# Import pre-trained CNN

```

```

from tensorflow.keras.applications import ResNet50V2
### 1. check how many layers are in the base model
### 2. define which layers will be fine-tuned
### 3. freeze the layers you don't want trained

# Model Construction

x = image_model.output
x = Conv2D(3, 6, activation="relu", padding = 'same')(x)
#x = Conv2D(1, 1, activation="relu")(x)
x = MaxPooling2D(pool_size=(2, 2), strides=(1, 1), padding
    = '
same')(x)
x = Conv2D(3, 6, activation="relu", padding = 'same')(x)
x = Conv2D(3, 6, activation="relu", padding = 'same')(x)
#x = Conv2D(1, 1, activation="relu")(x)
x = MaxPooling2D(pool_size=(2, 2), strides=(1, 1), padding
    = '
same')(x)
x = Conv2D(3, 6, activation="relu", padding = 'same')(x)
#x = Conv2D(1, 1, activation="relu")(x)
x = MaxPooling2D(pool_size=(2, 2), strides=(1, 1), padding
    = '
same')(x)
x = Flatten()(x)
x = Dense(36, activation = "relu")(x)
#x = Dropout(0.75)(x)
x = Dense(18, activation = "relu")(x)
#x = Dropout(0.75)(x)
x = Dense(9, activation = "relu")(x)
x = Dropout(0.3)(x)
predictions = Dense(n_classes, activation="sigmoid")(x)

print("Shotgunning beer...")

```

```

print("Model compiled, beer shotgunned, LET'S GOOOOOO")

model_final = Model(image_model.input, predictions)

# Compile the model
model_final.compile(loss = "categorical_crossentropy",
                    optimizer = optimizers.Adam(lr=0.0001)
                    ,
                    metrics=["categorical_accuracy"])

# Summarize model (make sure architecture looks what you
    thi
nk it looks like)

# MODEL TRAINING:
### 1. initialize the model
### 2. compile model
### 3. summarize model (make sure architecture is what you
    w
ant)
### 4. train model (save output in an object)
### 5. save model (avoid having to train and re-train)

# EVALUATE MODEL AND SAVE METRICS
### plot training and validation scores
### plot confusion matrices
### record Precision, Recall, and F1 score
### use data name, date, and time to save / index metrics
    ac
cordingly
### with the number of experimental treatments, it's
    importa
nt to be detailed and organized here

```


CHAPTER III

Multimodal Deep Learning for Detecting Election Administration Issues on Social Media

A growing body of election research uses social media data to augment information from national surveys. The volume of such data has spurred the use of machine learning to process different modes of content (text, image, audio, etc) found online. This paper uses deep learning to detect election incidents reported on Twitter during the 2016 U.S. presidential election. The study finds that such algorithms are more than capable of processing multimodal content, but the accuracy and prediction power of deep learning relies on the quality of the training data. The architecture introduced in this paper processes text and image data from thousands of tweets in minutes - and such models can easily be improved by increasing the scale of the training data.

3.1 Introduction

One metric to measure the quality of a democracy is to observe the ease with which its citizens vote. It is at the voting booth that citizens lift leaders to power and push legislative agendas forward. Obstacles to voting - like the presence of voter restrictions or the absence of polling place resources - indicate decay in the process. Election scholars continue to develop new methods to measure voter accessibility in the United States. *Election forensics* uses statistical analysis of data to determine whether an election’s outcome properly reflects the will of voters.

One challenge in this endeavor is determining whether anomalies in election data result from deliberate fraud (*Mebane Jr et al.*, 2018). Election issues can stem from logistical or administrative shortcomings, as opposed to direct fraud or tampering. Such shortcomings dampen voting efforts and distort turnout data, making it difficult to discern whether unanticipated results indicate more malicious threats to the electoral process. Contributing to election forensics, this study uses tweets collected during the 2016 U.S. presidential election to detect *election incidents*: anecdotal evidence of either a positive or negative voting experience.

Unfortunately, the United States does not have a single, national institution responsible for monitoring elections or handling voter complaints across the country. Although most states do have a system for processing election complaints, there are often multiple, prohibitively complicated avenues for a would-be complainant to pursue. As such, there is little aggregated data capturing voting experiences in the United States. Researchers have turned to alternative data sources, like the social media platform Twitter, to capture firsthand accounts of the voting process (*Abilov et al.*, 2021; *Wu and Mebane Jr*, 2020).

In this vein, this paper uses convolutional neural networks - called CNN’s or *deep learning models* - to process the text and images from thousands of tweets. The two models, one trained on text-only and the second trained on combined text-image, achieve high training accuracy on human coded training sets. Performance declines, however, when the models attempt to classify previously unseen validation and test data. More work must be done to increase the quality of training data, so machine learning models can better capture generalizable patterns indicative of election incidents.

Election Forensics Administrative shortcomings of interest to election researchers include long wait times and crowded polling places; poorly designed ballots; malfunc-

tioning or defective election equipment; and the availability of local polling places (*Stewart III and Ansolabehere, 2015; Herron and Smith, 2016; Lausen, 2008; Herrnson et al., 2009; Jones and Simons, 2012; Brady and McNulty, 2011*). Information capturing difficulties voters face on election day would be helpful, but is not readily available.

Election observation reports would seemingly be useful to researchers, as elections are now regularly observed by international monitors. The lack of international standards can lead to bias from election monitors (Hyde, 2011; Kelley, 2012). International monitoring is not the only source of observation data. Several countries have domestic institutions where citizens or political parties can formally contest election processes. Election data from Germany and Mexico reveal that statistics employed by election forensics are sensitive to the difference between strategic voter behavior and potential fraud (Mebane and Wall, 2015; Mebane, Klaver, and Miller, 2016). Being able to automatically detect voting experience would provide invaluable information in determining whether an election’s process has broken down. This study takes a step toward that end, using data from the United States.

As training data for the deep learning models, this paper uses tweets scraped from the Twitter Streaming API during the 2016 U.S. presidential election (*Mebane Jr et al., 2017*). One example is seen in Figure 3.1. The tweet illustrates a positive experience for one voter, who comments how well organized their polling place was in Clark County, Nevada. The “Vote Here Today” flag suggests the photograph was taken outside the polling place and the subject is holding a “Trump” sign, indicating partisan affiliation. To properly leverage all this information from text and image content – and potentially improve performance of automated classifiers – researchers have started using machine learning classifiers trained on multimodal input *Wu and Mebane Jr (2020)*. This study finds the promise of multimodal classification dependent on the quality and variation in the training data.

This paper is organized as follows: Section 2 details two election administration issues that are hard to detect using survey data alone; Section 3 discusses the Twitter data gathered during the 2016 U.S. presidential election, used as training data for classifiers that could remedy this problem; Section 4 reviews deep learning architecture and its application for multimodal tweet classification; 5 discusses training results and avenues for future research in this field.



Figure 3.1: A sample tweet showing a positive polling place experience: "just voted today , well organized by @ clarkcountynv . vote early or vote on election day ! its our duty as american ci & just voted today , well organized by @ clarkcountynv . vote early or vote on election day ! its our duty as american citizens . # leadright2016"

3.2 Election Administration Issues and Their Impact

As election administration researchers debate the impact voter ID laws and polling place resources have on turnout, one thing is clear: additional data sources are needed. While social media helps in this regard, researchers must be armed with the methods to handle millions of unstructured data points. This paper contributes two models that can handle both the scale and multimodal content of social media data.

Voter ID Laws Evidence suggests that the appearance and passage of restrictive voting laws are strategically employed by partisans in states with closer elections and a higher proportion of racial minorities (*Bentele and O'brien, 2013*). Voter ID laws suppress votes either directly or indirectly, but it is difficult to make this distinction using survey data alone. *Indirect voter suppression* occurs when voter ID laws intimidate citizens into not voting, even when they have proper identification. Potential voters might decide not to vote because they feel targeted by these laws; this is especially true among racial minorities, who have historically been subject to polling place violence. There is debate about whether or not voter ID laws actually suppress turnout among marginalized communities (*Hajnal et al., 2017; Grimmer et al., 2018; Chen et al., 2019*).

One contention is that "strict voter identification laws substantially alter the

makeup of who votes and ultimately skew democracy in favor of whites and those on the political right. These laws have a significant impact on the representativeness of the vote and the fairness of democracy” (*Hajnal et al.*, 2017). Another argument is that any estimated effect of voter ID laws on turnout is attributed to measurement error. National surveys like the Cooperative Congressional Election Study (CCES) are ill-suited for estimating the effect of state election laws on voter turnout because they are not representative of hard-to-reach populations (*Grimmer et al.*, 2018). Access to social media data gives researchers evidence of voter suppression, but the scale of such data requires salable methods. According to Twitter’s API documentation, users produce 500 – 700 million tweets per day. For reference, the two models introduced later in the paper are trained and validated on a corpus of less than nine thousand tweets.

Questions regarding the effect of voter ID laws revolve around national surveys, like the CCES, that lack the precision to capture voting patterns in groups most likely to be impacted by restrictions (*Grimmer et al.*, 2018). Election researchers continue to look for alternative information sources, as both cell phone and social media data have been used to examine wait times and experiences during the 2016 U.S. presidential election (*Chen et al.*, 2019; *Wu and Mebane Jr.*, 2020). Research suggests that mobile phones and social media provide a platform for racial minorities to voice political grievances (*Freelon et al.*, 2016). This study uses social media data to supplement information currently available to election scholars, aggregating first-person accounts of voting experiences posted to Twitter.

Polling Place Resources If time is a resource, then wait-time is a major cost voters must pay to cast their ballot. Research points to three factors that correlate with variation in wait-times: demographics, polling place operations, and the presence of voter ID laws (*Stein et al.*, 2020; *Cottrell et al.*, 2020). The danger with long lines is not only the increased cost for voters, but also the *perception* of high costs from *potential* voters. Line length has a negative effect on the number of people who arrive at a polling place and leave without voting because they see the line is too long (*Stein et al.*, 2020).

There is concern regarding the long-term effects that irregularities have on voter confidence and behavior. Difficulty at the polling place raises voters’ concerns that ballots were properly counted, eroding trust in the democratic system and making participation in future elections less likely. This *downstream* consequence of long wait times is well established. Findings indicate a linkage between long wait in the 2012

U.S. presidential election and turnout in 2016: "not only are some voters penalized by waiting in line, but a small number of these individuals appear to be dissuaded from voting in the future" (*Cottrell et al.*, 2020, 23). For every hour a voter waits in line, their probability of voting in the next election decreases by one percentage point (*Pettigrew*, 2020).

Further still, this downstream effect on voting behavior disproportionately impacts African-Americans over white voters (*Pettigrew*, 2020). For example, voters in Florida during the 2012 election who waited a long time to vote - and ultimately voted past the official 7:00 p.m. closing time - were disproportionately black, Latino, and Democratic (*Cottrell et al.*, 2020). The danger in this regard is that historically disenfranchised groups will grow more cynical as they face greater hurdles to the ballot box, eventually refusing to participate in elections altogether. Counter arguments suggests the dampening of voter confidence is limited in scope (*King*, 2020). The ability to automatically detect election incidents would illuminate voter sentiment toward the electoral process.

The first step in remedying polling place inadequacies is identifying them. Survey data has proved insufficient in this regard. The methods introduced in this paper take a step toward automatically detecting election incidents online. Large-scale, curated training data specific to political science research is still scarce, although there is progress in this regard (*Abilov et al.*, 2021; *Torres and Cantú*, 2021). To overcome this data shortage, I use *transfer learning* and pre-trained CNNs to extract features from a new data set - in this case, a corpus of tweets.

The use of deep learning models in this study aligns with the assertion that these methods are "a transparent, cost-efficient mechanism to record the information" necessary to validate the results in a national election (*Torres and Cantú*, 2021, 13). If deep learning can automate this process even partially, not only does this shorten the waiting time for results but it also increases voter confidence in the election process itself. This study contributes to computational social science by exploring deep learning's potential to detect election administration issues on social media. Perhaps most importantly, this paper demonstrates that deep learning is flexible enough to classify multimodal content. Future election researchers can use these methods across a variety of classification problems and data types.

3.3 Twitter Scraping During the 2016 U.S. Election

This section details data collection efforts during the 2016 U.S. presidential election (*Mebane Jr et al.*, 2017). The purpose of the original study was to capture voting problems posted to Twitter and use geolocation to identify "hot-spots" of polling place irregularities across the United States. The authors used two platforms to gather Twitter data over the course of the 2016 election cycle: the Sysomos MAP archive and the Twitter API. The Sysomos MAP search tool was used to gather data during the primary/caucus period because the platform supports keyword searches for tweets going back 12 months (data collection efforts began in the summer of 2016, after the primaries had been decided). The Twitter API was used to gather data during the general election period. Note that this actually consists of two different interfaces: the REST API for downloading past tweets and the Streaming API for downloading tweets as they are posted in real-time.

The motivation for using Twitter was the inaccessibility of voter complaint data from election administration officials across the United States. There is no one central or national agency responsible for processing polling place irregularities reported by voters. Individual states may have such departments, but the process of reporting complaints can be so convoluted it dissuades voters from doing so, resulting in little data ever being recorded (*Mebane Jr et al.*, 2017). That said, many of these departments have a social media presence. The timelines of 493 Twitter accounts from the REST API were downloaded using a list of election official, party and other Twitter account names ("twitter handles"). Meanwhile, the Twitter Streaming API enabled the downloading of tweets as they were posted to Twitter in real-time using keywords as filters.¹ From October 1 through November 8, 2016, 44,329 tweets were downloaded from timelines and 16,221,304 tweets from the Streaming API.

Retweets were removed, leaving 6,163,890 unique tweets with 4,541,097 unique

¹Twitter API Keywords: line to vote, long line to vote, wait to vote, absentee voting, early voting, problems voting, voting rights, right to vote, election fraud, corruption, voter fraud, stole election, stolen election, rigged, election stealing, tamper, manipulate, voter id, voter identification, election complaint, election problem, broken voting machine, election officials, electronic voting, election audit, election observer, poll watch, vote protection, election protection, disenfranchised, campaign finance, election system, primary election, general election, voter complaint, polling place, registration database, statevote, votestate, stateelection, vote count, vote tabulation, voter database, voter registration, voter suppression, voting machine, voting machine hacked, vote not counted, vote, US election, American election, not enough ballots, absentee ballot, voter intimidation, voter harassment, mail in ballot, vote by mail, voter hotline, waiting to vote, precinct, precinct officials, precinct captain, replacement ballot, ballot selfie, my ballot, my vote, eleccion, fila para votar, derecho al voto, derecho al votar, fraude electoral, maquina de votacion, funcionarios electorales, colegio electoral, neo-nazi, white supremacist, white nationalist, alt-right.

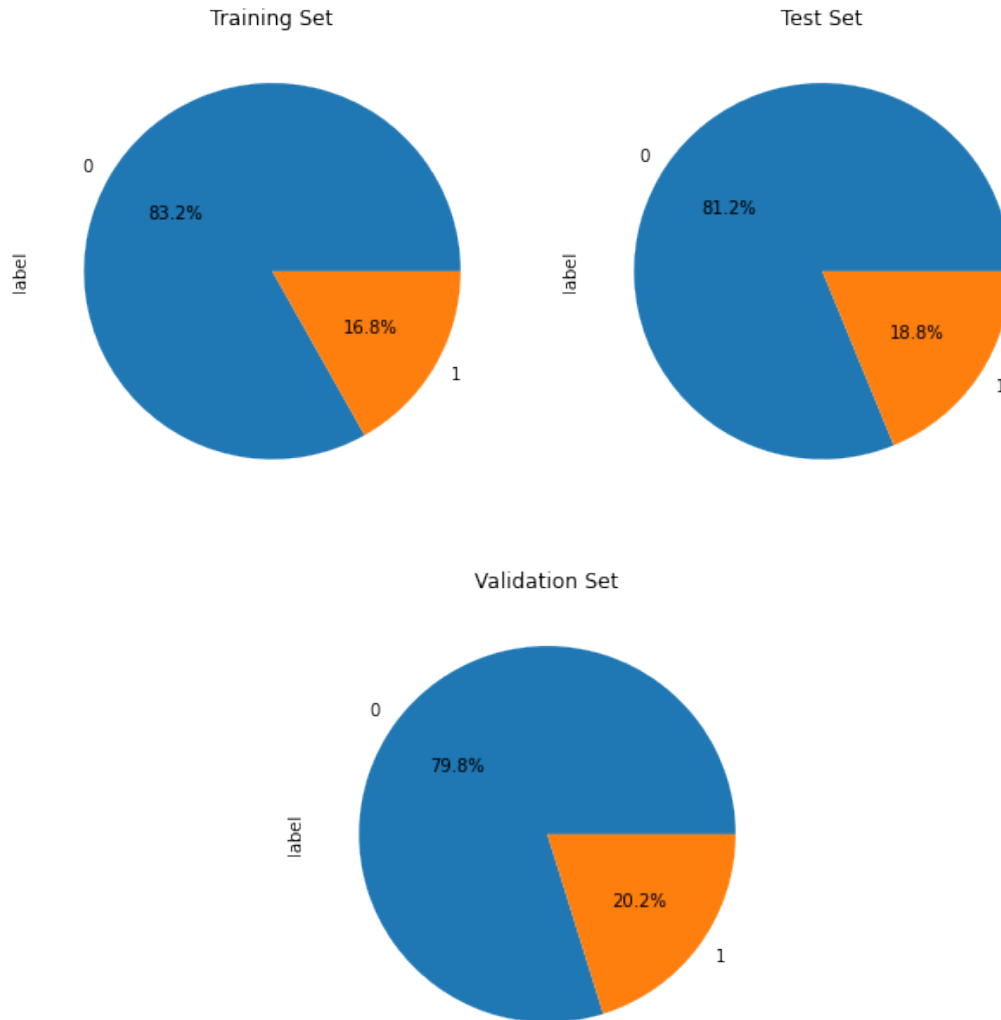


Figure 3.2: *Class proportions across the training, validation, and test sets.* The original data is subset, with the ratio of non-incidents (0) to incidents (1) preserved. The bulk of the data (n=7950) is used for training, while the validation set (n=500) is used to check for overfitting and the test set (n=500) is used to gauge the final model’s predictive power.

tweet texts. As was previously mentioned, the goal of the initial study involved locating incident observations – at the state, city or neighborhood levels – but not all tweets have *geotags* associated with them. Twitter users must change their privacy settings to have their location automatically recorded at the time a tweet is posted. As such, only the 598,783 tweets that had geographic information were kept, 505,112 of which contained unique tweet texts. A sample of 19,789 tweet texts were taken and labeled by hand as containing an incident observation ($n = 2,610$) or not ($n = 17,179$). About 8,500 of these coded tweets featured accompanying images that were downloaded after the tweets were originally pulled from the Twitter API. This sample of 8,500 are used as the training data for the election incident model.

The intuition behind the *supervised learning* approach is to provide the deep learning model with enough training examples so that its weights can detect important features in unseen images and accurately predict their labels. For the sake of parameter tuning and testing, the hand-coded training data is split into three subsets, as shown in Figure 3.2. *Training* data ($n=7950$) gets input into the model so it can learn major features and optimize weights. To check for overfitting, the model attempts to classify unseen *validation* data after each iteration of training (called an *epoch*). Once parameter tuning is completed, *test* data provides performance benchmarks for the final model. Note that the subsets are sampled such that the ratio of *non-incidents* (0) to *incidents* (1) is preserved across the three samples.

3.4 Deep Learning for Tweet Classification

This section provides an overview of the deep learning methods used to detect election incidents. Convolutional neural networks are layered structures for processing and classifying high-dimensional data (*Hastie et al.*, 2009). CNN’s are considered *feed-forward networks* because data is passed forward from the input layer to the middle and output layers. An example of this process, taken from image layers of the combined text-image model, is shown in Figure 3.3. Training a CNN requires the use of weights that differentiate between relevant features in the input data; weights and input data are combined via matrix multiplication before being fed forward to the next layer. The network performs different operations on the data to capture important signals, reduce dimensionality, and find relationships in the data.

There are two phases for a CNN to process content: feature extraction and classification. During feature extraction, *convolutional* layers use filters that rotate across the data, searching for specific patterns and breaking down each individual input

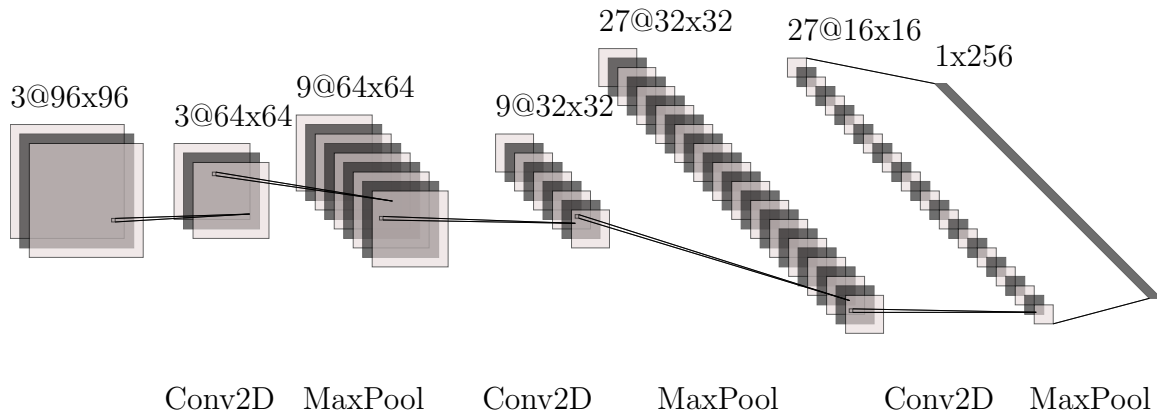


Figure 3.3: *Schematic of the image layers from the text-image model.* Convolutional filters create feature maps while pooling layers reduce the size of the data. This visual is specific to the image side of the architecture. The text side is parallel, with one-dimensional matrix operations instead of the two-dimensions required for images.

into *feature maps*. *Pooling* reduces the dimensionality of the data by either taking the average or the maximum among clusters of data points, reducing noise in favor of the strongest signals.

The full architecture of the combined text-image classifier is shown in Figure 3.4. Between the convolutional and pooling operations, a lot of effort is expended reducing images to their most essential features, making data easier to process in the final layers of the network. After feature extraction, the final layers of a CNN is concerned with classification. Feature maps are fed into the *dense* layers – where all of the inputs from one layer connect to all functions in the subsequent layer. This enables the network to detect non-linear relationships like image patterns or keywords that suggest a tweet’s membership in one class versus another.

3.4.1 Transfer Learning

Spurred by the costliness of training complex CNN’s, machine learning research has explored transfer learning – a technique where a model’s parameters, trained on a *source task*, is applied to a new *target task*. Transfer learning defies the assumption that training and target data must come from the same domain; often, researchers have a target task in one area of interest but use a model pre-trained on data from a separate, more general domain (*Pan and Yang, 2010*). This process works well with image data because images share a feature space. The lower levels of a neural network learn basic patterns, like edges and shapes, that are not exclusive to any particular domain, but rather, common to all pictures. Transfer learning seeks to improve

performance by using the parameters of a pre-trained model to extract features from the target data. More formally:

Definition 2. Transfer Learning. *Given a source domain D_S and learning task T_S , a target domain D_T and learning task T_T , transfer learning aims to help improve the learning of target predictive function $f_T(\cdot)$ in D_T using the knowledge in D_S and T_S , where $D_S \neq D_T$, or $T_S \neq T_T$.*

Let domain D be defined by a *feature space*, X , and a marginal probability distribution, $P(X)$. For a domain of interest, $D = \{X, P(X)\}$, a task, T , has two parts: a set of labels Y and a predictive function $f(\cdot)$. A task simply refers to a function that predicts labels: $T = \{Y, f(\cdot)\}$. Much like research assistants, the predictive function learns from examples. This training data consists of pre-labeled data $\{x_i, y_i\}$ where $x_i \in X$ and $y_i \in Y$. The ultimate goal is to estimate the parameters of $f(\cdot)$ that can predict the label $f(x)$, given a new instance x .

Assume there are only two domains of interest: a source domain D_S and a target domain D_T . Data sampled from the source domain data is denoted $D_S = \{(x_{S_1}, y_{S_1}), \dots, (x_{S_n}, y_{S_n})\}$ where $x_{S_i} \in X_S$ is a data point and $y_{S_i} \in Y_S$ is the corresponding label. Equivalently, the target domain data is denoted $D_T = \{(x_{T_1}, y_{T_1}), \dots, (x_{T_n}, y_{T_n})\}$ where $x_{T_i} \in X_T$ is the data point and $y_{T_i} \in Y_T$ is the corresponding label. Transfer learning is the process of applying knowledge gained from D_S to a task in D_T . This study uses parameters from three different pre-trained networks. The text-only classifier makes use of GloVe word embeddings, pre-trained parameters that measure word similarity via Euclidean distance (more below) (Pennington et al., 2014). As seen in Figure 3.4, the text-image classifier makes use of these embeddings on the text side and the VGG16 model on the image side - a very deep CNN whose frozen parameters do not update, but extract features from tweet images (Simonyan and Zisserman, 2014).

Model Architecture After the images pass through the VGG16 model, their feature maps are fed to adaptation layers, whose weights are updated over the course of training. These layers perform various operations on the features to reduce dimensionality and capture relevant patterns. Operations for both the text and image side of the model include alternating convolutions and max pooling. Features are concatenated then fed to X dense layers. The adaptation layers of the text-image model consist of 26,475 trainable weights that update and learn patterns in the feature data; the text-only model consists of 21,250 trainable weights. Although this seems like a

small discrepancy, proportional to the size difference of the data, note that the text-image model uses 14 million frozen weights for feature extraction, while the text-only model uses just over 600,000 frozen weights. For further details on image processing and model architecture, please see the code overview in Appendix I of this chapter.

Parameter Tuning The overall number of layers, the size of the convolution filters, the type of activation function, and the type of pooling are all *tuning-parameters* – adjustable settings defined by the user that impact both the speed of training and the accuracy of the model. An enormous amount of time and energy is spent testing different combinations of tuning-parameters to find what configuration provides the best performance. The difficulty in parameter tuning is that there are no hard and fast rules about how to determine the optimal configuration; no two networks are built the same, as configurations are specific to the problem statement and classification scheme. There are certain heuristics that can guide experimentation (for instance, it is generally best to increase depth depending on the complexity of the image) but this requires trial and error.

Parameter tuning is a balance between potential improvements and costs to model performance. Increasing the number of filters helps improve accuracy, but this requires more training data. Increasing the size of the learning rate helps the optimization algorithm converge faster, but runs the risk of unstable training and sub-optimal parameters. A learning rate that is too small will result in a long training process that gets stuck at a local minimum. Too many filters in a model runs the risk of overfitting – when a model adjusts parameters to accurately classify training data without detecting the relevant patterns that let it classify unseen data. An example of overfitting can be seen in the training and validation metrics of the text-only model, shown in Figures 3.5 and 3.6(b). When transfer learning is employed, as it is in this study, the choice of what pre-trained model to use is itself a tuning parameter. The GloVe and VGG16 model were chosen for transfer learning because they outperformed other models available in Python’s Keras package.

Backpropagation During training, optimal weights are found via a process called *backpropagation*. This process occurs in three stages – the forward pass, the backward pass, and the gradient update. During the forward pass, the linear combination of weights and input data is computed, then fed forward layer-to-layer until the model outputs the predicted label. During the backward pass, the distance between the predicted and true labels is calculated using the *loss function*. The derivative of

the loss function with respect to each weight, the *gradient*, is computed layer-by-layer starting from the output layer and going backward. The third step in backpropagation uses the calculated gradient to adjust the weights toward the steepest decrease of the loss function. This process of iteratively computing the gradient of the loss function with respect to each weight and then adjusting weights to minimize loss, called gradient descent, efficiently decreases error at each training step.

Benchmarks for Training Training performance is measured using two benchmarks: accuracy and cross-entropy loss. Accuracy, defined formally in Equation 3.1 below, captures the percentage of images the model correctly classifies. We define y_i as the true value for data point i and \hat{y}_i as the model’s predicted label. The higher the accuracy, the better the model.

$$\text{accuracy}(y, \hat{y}) = \frac{1}{n_s} \sum_{i=0}^{n_s-1} \mathbf{1}(\hat{y}_i = y_i) \quad (3.1)$$

Categorical cross-entropy loss, defined in Equation 3.2 below, is calculated as a sum of separate loss for each class label per observation. As probability p diverges from the true label, cross-entropy increases – loss closer to 1 indicates poor performance, while a perfectly performing model would have a loss of 0.

$$\text{loss}(y, p) = - \sum_{c=1}^M y_{i,c} \log p_{i,c} \quad (3.2)$$

In the above equation, we define M as the number of classes, p as the predicted probability, and $y_{i,c}$ is a binary indicator (0 or 1) that indicates if c is the correct class. The algorithm uses the loss function to capture prediction error. Gradient descent then updates weights to minimize loss, thereby reducing error before attempting to classify another batch of data. Training neural networks is thus framed as an optimization problem: update weights to reduce error until you arrive at a global minimum.

GloVe word embeddings and text processing Originally introduced by *Pennington et al.* (2014) GloVe is an unsupervised learning algorithm for representing words as vectors in space. Word embeddings are a natural language processing technique where words are cast into a geometric space such that distances between words capture their semantic similarity. The closer the words in space, the more similar they are to one another. Representing words as these global vectors means represent-

ing words by their semantic neighbors in geometric space. The location of that word in the space is referred to as its *embedding*. The algorithm learns global word-word co-occurrence statistics from Wikipedia, and the resulting vectors represent linear relations between words.

Image Processing To better understand how a CNN processes image data, think of an image as a three-dimensional matrix consisting of pixel values. Images are represented by their pixel values, from 1 to 256, along three color channels – red, green, and blue (RGB). Data is normalized by dividing all values by the range (255). This ensures that all of the input data exists on the [0,1] scale which enables the network to converge faster during training.

During training, the CNN performs a series of matrix transformations on the data, so there cannot be any variation in dimensionality of the inputs. The training algorithm expects all images to have a square, uniform size, so all images are resized to 96 x 96. For data augmentation, pre-processing images can include transformations to provide the model more variation from its training input. Training images can be randomly flipped, zoomed-in, or tilted along different axes. This provides the model with examples that are better representative of the feature space, improving classification accuracy of unseen data.

3.5 Results and Discussion

Both models perform well on training data, but poorly on validation and test data. Indeed, as seen in Figure 3.5, the text-only model eventually classifies the training set perfectly - note the high accuracy and low cross-entropy score - but the accuracy and loss metrics diverge from the training set. This indicates that the CNN's are *overfitting* the training data. This occurs when models become overly complex, when there are more parameters than observations, or when there is not enough variation in the training data for the model to detect generalizable patterns. Over the course of learning, the CNN's become overly sensitive to changes in the training data that are less prominent or meaningful in the validation and test sets. The remedy for overfitting is more data with more distinct classes and identifiable features.

Confusion matrices help researchers understand where models can be improved by gauging predictive performance on the test set, across classes. Figure 3.6 demonstrates that the poor predictive performance for the text-only algorithm comes from the model only correctly predicting 60% of *incidents* - the other 40% being mistakenly

classified as *non-incidents*.

The text-image model does not perform any better on classifying unseen tweets from the 2016 Election Twitter data. As Figure 3.7 indicates, the model achieves higher accuracy and lower cross entropy score on the training data, relative to the validation set. While the disparity is not as great as in the text-image model, the confusion matrices in Figure 3.8 show that the text-image model has trouble predicting *incidents* even on the training data. Although multimodal content classification remains an important goal for the analysis of social media data, not all classification schemes are going to benefit from multimodal input.

One potential claim might be that the parameters in the text-image model are merely suboptimal - perhaps the model requires more time to train or the filters need to be bigger - but no amount of parameter tuning can make up for inadequacies in the data. Indeed, the classes are incredibly imbalanced. There simply are not enough examples of polling place incidents in the 2016 Election data for the text-image classifier's parameters to detect a generalizable pattern. To avoid criticisms of suboptimal parameter tuning, future studies might adopt a method called grid searching - where tuning parameters are set at random and multiple models are training and compared until optimal parameters are found. As it is, such a practice is computationally intensive and beyond the scope of this paper.

The advantage of deep learning is its flexibility and scalability. Indeed, the text-only model processes tens of thousands of tweets in seconds, the text-image model does the same in minutes. Both models show the ability to replicate human coded classification schemes. Note that the tweets in the training sample were scraped from the Twitter Streaming API, which offers a 1% sample from the 500-700 million tweets written per day. Future research should aim to increase the scale of training data by accessing the 10% Decahose API which increases the variation in URL's, hashtags, and news topics (*Li et al.*, 2016).

Computational social science has already leveraged machine learning to automatically verify part of the election process, using a CNN to code vote tallies in Mexico's 2015 federal election (*Torres and Cantú*, 2021). This intersection between political science and machine learning is not without difficulty. The benchmark deep learning models used throughout machine learning literature are trained on generic data sets which have little bearing on politics. As election forensics moves forward, the larger goal of this work remains in focus: observing elections can help measure the quality of democracy. If deep learning helps us detect issues in election administration, then curating data sets and training deep learning models should be seen by political

scientists as worthwhile endeavors.

Performance could be improved by combining the efforts of the text-only and text-image models. Currently, all tweets in the data set contain images; for the sake of comparison, both models had to be trained on identical training sets. Only 8,500 of the 19,789 coded tweets contained images, so the algorithms are only being trained on a fraction of available knowledge. Note that the combined text-image classifier did not outperform the text-only model partly because the text-only model performed so highly on the training set.

To improve predictive power, the simplest way forward would be to re-build the system to account for tweets that are and are not accompanied by an image. Tweets that do not contain an image are fed to the text-only model; tweets that do contain an image are fed to the text-image combined model. Both models are trained, parameters tuned, until predictive performance is maximized (as evidence by the validation data). Once the models are trained on their respective data sets,

Shapley values can identify which images most contributed to error and accuracy. Taken from cooperative game theory, the solution concept traces how much the actions of any given player contributed to overall result. In this context, it presents the images that contributed most to changes in training accuracy. This information can help guide feature selection as training sets are curated.

Currently, this system could be deployed on election day to detect incidents in real-time; however, because of the error shown during validation, it could only be done in a semi-automated way. The classification system would inevitably overestimate the number of reported incidents. To correct for this, researchers could grab tweets on election day using a set of keyword filters, feed them to the classifier for an initial estimate of incidents, and then manually review the results to correct the algorithm's overestimation. Final results would be delayed, but the system would at least enable researchers to detect potential election fraud in-real time. Timeliness is important for in this regard. If there is no evidence of fraud, the algorithm would either reassure public confidence in the electoral process. Conversely, if Twitter is filled with anecdotes of administrative breakdowns, the algorithm would grab the relevant evidence for further investigation.

When mapping from concept to data, defining features of importance are just as important as defining the concept itself. The effort to manually label the initial training data was done in an haphazard way; that is, a lot of debate focused on how to define the concept *election incident* in the context of the Twitter corpus. With too many differences in opinion, the concept lacked cohesion in the training data. Little

attention was focused on what visual features were vital to this mapping. Curating a training corpus requires clearly defined concepts, but more importantly, patterns that are generalizable.

The machine must be exposed to visual and textual themes that it will see elsewhere in the data. A tweet being an election incident is a necessary condition for inclusion into the training corpus; but it should not be a sufficient condition. A training corpus should include examples that the machine is likely to see *beyond the training data*. If a tweet is an election incident – but only weakly so – it is likely adding to the noise instead of strengthening the signal. Machine learning algorithms optimize parameters to detect recurring patterns specific to each class. If there is too much variation in these patterns – if one of the classes is too broad, its components only loosely connected – there will not be enough information for learning to gain traction.

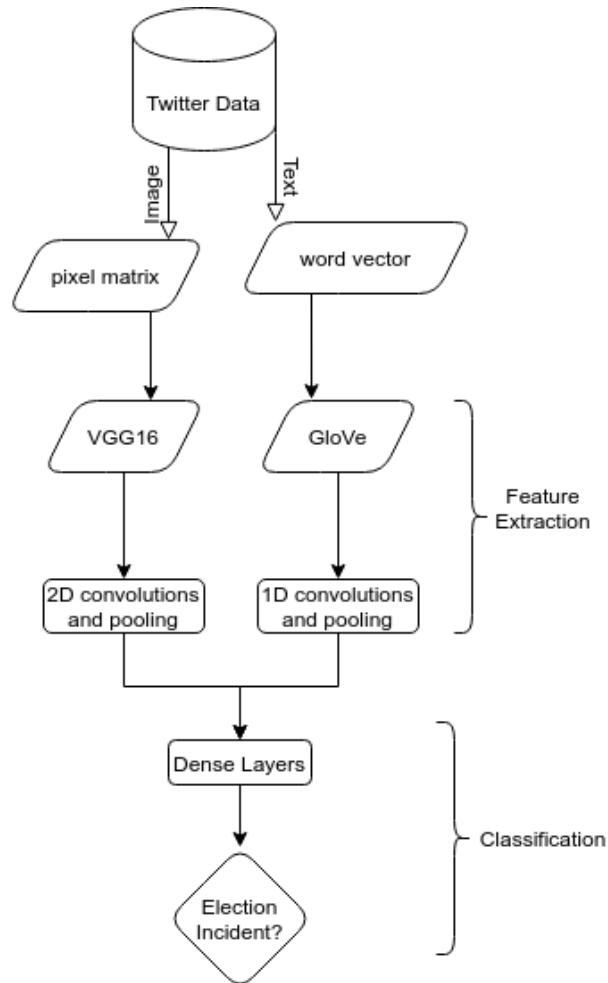


Figure 3.4: *The text-image model’s parallel architecture.* Content on both sides is represented in matrix form. A combination of transfer learning (from the VGG16 image classifier and GloVe word embeddings), convolutions, and pooling layers extract features from a tweet’s content. Once their features have been joined, dense layers learn patterns from both text and image features, before making a classification decision.

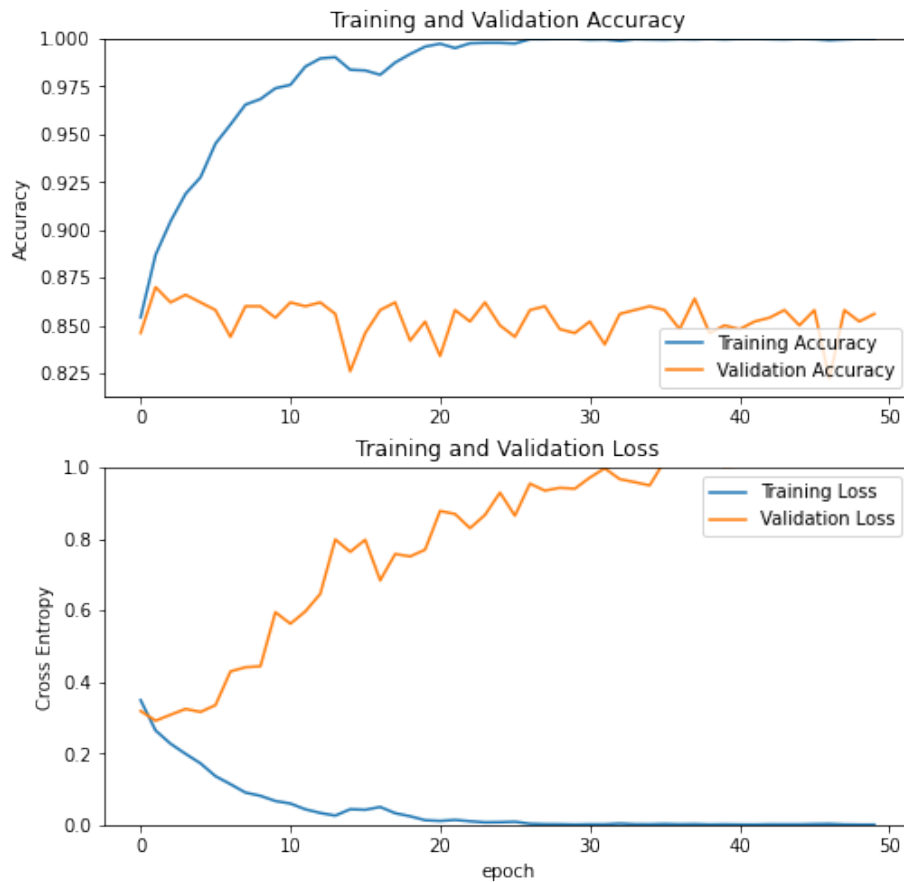


Figure 3.5: *Training and validation metrics for the text-only model.* The most striking element about the text-only model is not its high training accuracy, but rather, the disparity between training and validation metrics. The model suffers from *overfitting*. The remedy for this is to code more training data so the classes become more distinct, their features more identifiable by the algorithm.

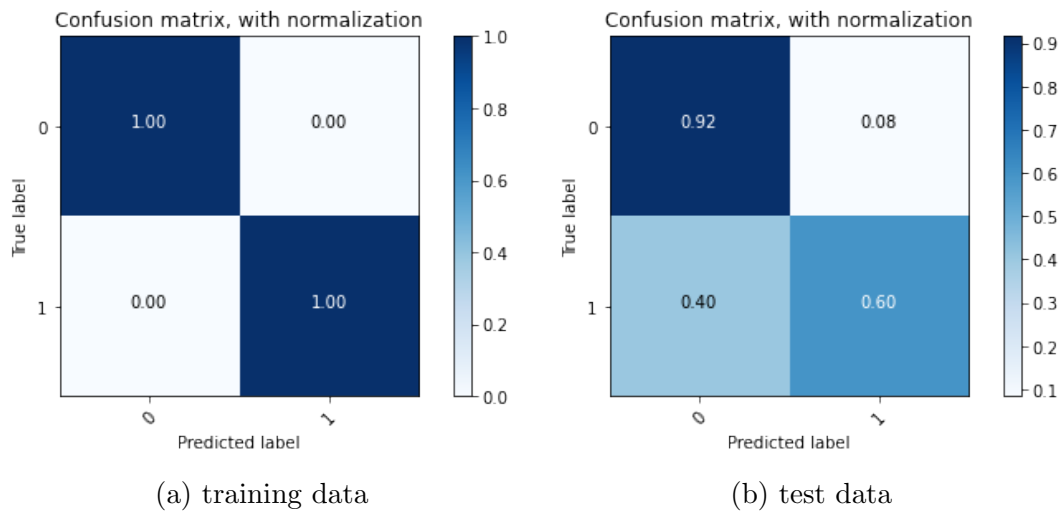


Figure 3.6: *Confusion matrices for the text-only model.* There is a significant performance decline when the text-only algorithm attempts to classify previously unseen data. The model learns to perfectly classify tweets in the training data (a) but only manages to detect sixty percent of election incidents in the test set (b).



Figure 3.7: *Training and validation metrics for the text-image model.* The text-image classifier does not perform as well as the text-only classifier on the 2016 Election Twitter data. Although multimodal content classification remains an important goal for the analysis of social media data, not all classification schemes are going to benefit from multimodal input.

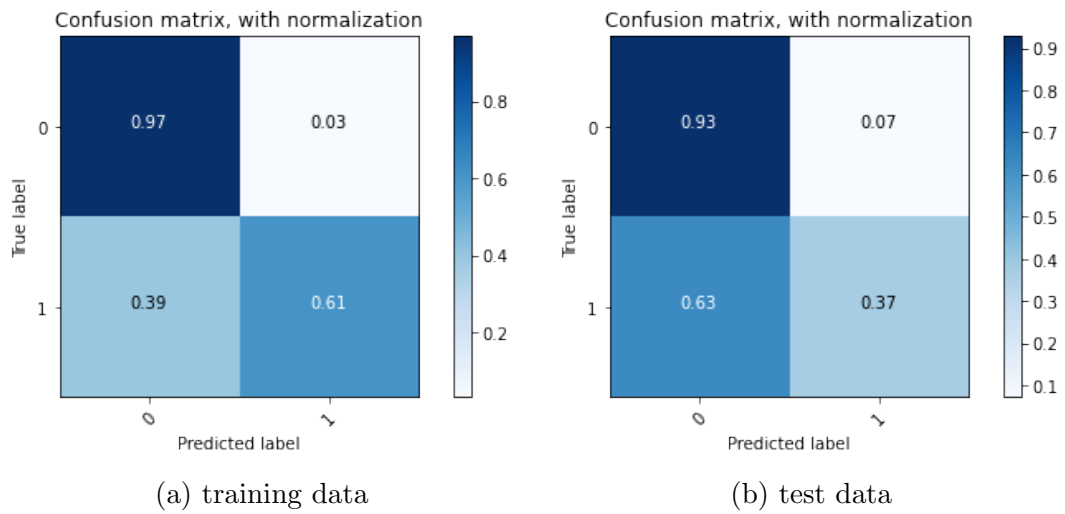


Figure 3.8: *Confusion matrices for the text-image model.* The poor performance on the training data indicates that a multimodal classifier would require a larger number of *incidents* in the training data to properly distinguish between features.

3.6 Appendix I: Code Overview

```
#!/usr/bin/env python
# coding: utf-8

#####
##### I. DATA PRE-PROCESSING #####
#####

# Pseudo-code for Text and Image Tweet Classifier:
    Election
Forensics Model
DOI: 10.13140/RG.2.2.35225.49763

# For full code, visit:
https://github.com/apineda91/DeepLearn4PolComm
python libraries required: tensorflow, numpy, sklearn,
    nltk,
pandas, keras (part of tensorflow), matplotlib, os, sys

# Define function to plot_confusion_matrix
def plot_confusion_matrix

### 1. define columns and load data
### 2. make sure 'label' var is a string
### 3. check sizes of each class
### 4. drop null values
### 5. make sure sample size is divisible by batch size
### 6. shuffle data

cols = ['tweet_id', 'date', 'text', 'support', 'hashtags',
    ,
users', 'urls', 'media_urls', 'nfollowers', 'nfriends', '
    fil
e_name', 'path']
```

```

DATA = pd.read_csv('./elections_june21.csv', dtype = {'
    doc_i
d':str, 'picfiles':str,'label':str})
DATA = DATA.sample(frac=1, random_state=1234).reset_index(
    dr
op=True)

# Define image size
# Define batch size
# Define path to images
# Check that the images
    # 1) exist
    # 2) are where you think they are

DATA = DATA[[os.path.isfile(i) for i in DATA['path']]]

# Split data into the following:
### 1. Validation (to check for overfitting)
### 2. Test (set aside until final model evaluation)
### 3. Training (for training)

### spot check class proportions to make sure they're
    compar
able across sets ###

##### Define data generators that can handle multimodal
    inpu
ts #####

def text_generator(a,labs, n):
    while True:
        for i in range(a.shape[0] // n):
            d2 = a[n*i:n*(i+1)]
            y_text = labs[n*i:n*(i+1)]
            yield d2, y_text

```



```

def multi_input_generator(df, x_image, y_image, x_txt,
    y_txt
, b_size):
    t1 = input_imgen.flow_from_dataframe(
        dataframe=df,
        directory = './labim_all',
        x_col = x_image,
        y_col = y_image,
        target_size=(img_width, img_height),
        batch_size=b_size,
        class_mode='categorical',
        validate_filenames=False)

    t2 = text_generator(a=x_txt, labs = y_txt, n=b_size)

    while True:
        d1,y = t1.next()
        d1 = np.expand_dims(d1, axis = 0)
        d2, y_text = t2.__next__()
        yield [d2, d1[0]], [y_text, y]

##### i. TEXT PRE-PROCESSING #####
### 1. Define hyper-parameters
### 2. Subset to only data and labels
### 3. Tokenizers for training text data
### 4. Tokenizers for val text data

##### Initialize the following data generators #####
### 1. train_generator
### 2. val_generator
### 3. test_generator

##### PREPARING GLOVE LAYER #####
### 1. build index mapping words in the embeddings set
to their embedding vector
### 2. compute and prepare embedding matrix

```

```

### 3. load pre-trained word embeddings into an Embedding
    la
yer

```

```

##### ii. IMAGE PRE-PROCESSING #####

```

```

### 1. load pre-trained CNN
### 2. check how many layers are in the base model
### 3. define which layers will be fine-tuned
### 4. freeze the layers you don't want trained

```

```

#####

```

```

##### II. MODEL CONSTRUCTION #####

```

```

#####

```

```

sequence_input = Input(shape=(None, ), dtype='int64')
embedded_sequences = embedding_layer(sequence_input)
x_text = Conv1D(32, kernel_size = 5, activation="relu",
    padd
ing = 'same')(embedded_sequences)
x_text = MaxPooling1D(5)(x_text)
x_text = Conv1D(32, kernel_size = 5, activation="relu",
    padd
ing = 'same')(x_text)
x_text = GlobalMaxPooling1D()(x_text)
x_text = Dense(16, activation="relu")(x_text)
#x_text = Dropout(0.5)(x_text)
preds = Dense(2, activation='relu')(x_text)

```

```

# Adding custom Layers

```

```

x_image = image_model.output

```

```

x_image = Conv2D(3, 1, activation="relu", padding = 'same
    ')(

```

```

x_image)

```

```

x_image = MaxPooling2D(pool_size=(2, 2), strides=(1, 1),
    pad

```

```

ding='same')(x_image)
x_image = Conv2D(3, 1, activation="relu", padding = 'same
')(
x_image)
x_image = MaxPooling2D(pool_size=(2, 2), strides=(1, 1),
pad
ding='same')(x_image)
x_image = Conv2D(3, 1, activation="relu", padding = 'same
')(
x_image)
x_image = MaxPooling2D(pool_size=(2, 2), strides=(1, 1),
pad
ding='same')(x_image)
x_image = Flatten()(x_image)
img_predictions = Dense(2, activation="relu")(x_image)

merged = Concatenate()([preds, img_predictions])

# We stack densely-connected network on top
x = Dense(64, activation='relu')(merged)
x = Dense(32, activation='relu')(x)
x = Dense(16, activation='relu')(x)
x = Dense(8, activation='relu')(x)
x = Dense(4, activation='relu')(x)
#x = Dropout(0.5)(x)
main_output = Dense(2, activation='sigmoid', name = '
main_ou
tput')(x)

# Defining a model with two inputs and one outputs
elections_model = Model([sequence_input, image_model.input
],
[main_output])

tbCallBack = TensorBoard(log_dir='./Graph', histogram_freq

```

```

    =0
, write_graph=True, write_images=False)

print("ROMA VICTOR!")

sgd = optimizers.SGD(lr=0.01, decay=1e-5, momentum=0.95,
    nes
terov=True, clipvalue=.5)
elections_model.compile(optimizer=sgd, loss="
    categorical_cro
ssentropy", metrics = ['categorical_accuracy'])

# MODEL TRAINING:
# 1. initialize a model with two inputs and one outputs
# 2. initialize optimizer
# 3. compile model
# 4. train model (save output in an object)

#####
##### III. EVALUATION AND VALIDATION #####
#####

# For confusion matrix, we need to compute predictions
with
our trained model
# 1. Grab training predictions
    # convert to an array of binary values
# 2. Grab validation and testing predictions
    # convert to arrays of binary values
# 3. Input predictions and true values into
    plot_confusion_m
atrix

```

CHAPTER IV

Multimodal Deep Learning for Capturing Attitudes Toward Black Lives Matter

Capturing attitudes toward the Black Lives Matter movement provides scholars with deeper insight into public opinion on racialized in the United States. Twitter data is useful in this regard, but the scale and scope of such data can be prohibitively large – for instance, data for this study comes from a corpus of 40 million tweets. Manually coding that many tweets is not a feasible undertaking. Recent social movement research has sought to automate the analysis of social media data, focusing on sentiment of tweet text. This methodology is limiting because it leaves overall attitude ambiguous – further it ignores the importance of visual data in expressing political opinions online. This paper uses convolutional neural networks – called *deep learning* models – to classify tweets based on their text *and* image features. Results indicate the inclusion of image features helps the model overcome imbalances in the training data. More work must be done to improve the quality of training data available for computational social scientists employing machine learning methods.

4.1 Introduction

I am America. I am the part you won't recognize. But get used to me. Black, confident, cocky; my name, not yours; my religion, not yours; my goals, my own; get used to me.

Muhammad Ali, 1970

When Alicia Garza penned *Love Letter to Black Folks* in 2013, she ended the essay with the assertion: "We need to love ourselves and fight for a world where Black lives matter." The letter was written in response to George Zimmerman's acquittal, following his shooting and killing of seventeen-year-old Travon Martin. Beyond sparking a social movement, Ms. Garza's assertion that *Black lives matter* sparked debate over the role of racial frames in discussions of police violence. As the Black Lives Matter (BLM) movement grew – via offline protests and online dialogue – as did conversations surrounding BLM protests of police violence. Studying these online conversations can help social movement scholars gain an understanding of the relationship between protest activity and public opinion.

There is debate among scholars over the effectiveness of social movement protests in making lasting change on public opinion. Studying attitudes toward BLM would help clarify this point, providing insight into public opinion toward racialized issues. Just as social movements evolve and change over time, so do attitudes toward movements and their protest activity. Following the 2020 wave of protests, BLM successfully shifted public attention toward antiracist issues (*Dunivin et al.*, 2022). This research fails to account for variation in attitudes toward the movement – not all attention is positive attention. Data shows that any shifts in public opinion as a result of the 2020 BLM protests were temporary and negligible (*Reny and Newman*, 2021). This line of research would benefit from more nuanced measurements of public opinion, especially as it relates to social movements. This paper contributes in this regard, using deep learning algorithms and Twitter data to differentiate between tweets that support and oppose the Black Lives Matter movement.

The importance of social media data in studying movements cannot be overstated. Movements uses social media to spread information, raise awareness, and engage public discourse; social media platforms are where coalitions are built and meaning is

collectively cultivated (Bonilla and Rosa, 2015; Mundt et al., 2018). Twitter data is especially useful in capturing the motivation *behind* attitudes, providing a deeper understanding of shifts in opinion. The scale of social media data creates methodological problems for social scientists. Recent research into Black Lives Matter employs Twitter data, but focuses solely on sentiment of tweet text (Patnaude et al., 2021). In this context, a tweet’s sentiment could relate to either police violence *or* to the social movement *opposing* police violence – overall political attitude is left ambiguous. Political discourse online is exceedingly complex and nuanced. If movements like Black Lives Matter, MeToo, and the Arab Spring use social media sites to start a conversation, social movement and public opinion scholars should equip themselves with the computational tools necessary to study that conversation.

This paper uses deep learning to build an algorithm capable of learning on text and image data. This enables the algorithm to consider the entirety of a tweet’s content when deciding whether it signals support or opposition toward BLM. Findings indicate that image content improves the classifier’s ability to predict previously unseen tweets. These types of machine learning algorithms enable researchers to keep pace with dynamic, contemporary social movements. Social media data helps capture the public’s short-term reactions protest activity; when aggregated over time, such data would also give scholars insight into long-term shifts in public opinion toward social movements.

The rest of the paper is organized as follows. Section 2 discusses the importance of capturing public opinion as it relates to social movements. Section 3 details data gathering efforts on Twitter and shows examples of tweets supporting or opposing BLM. Section 4 reviews deep learning methods and the architectures built for classification. Section 5 details training results, in addition to avenues for future research.

4.2 Capturing Public Opinion Toward BLM Protests

Studying attitudes toward Black Lives Matter reveals larger trends in public opinion toward racialized issues. These attitudes are “indicative of what people believe police violence, and protest against it, tell us about the state of democratic society” (Leach and Teixeira, 2022, p. 4). Negative attitudes toward civil rights issues can reveal racial prejudice against the Black community, or Black protest activity (Bobo, 1988). Critics of Black Lives Matter attribute their attitudes not to racial animus, but rather, to an unwavering support of the police. Evidence from survey data contradicts this claim: “the primary motivation for white opposition to BLM was not

support for the police but was instead an animus toward Black Americans and commitment to a racial logic that justifies white privilege” (*Drakulich et al.*, 2021, 244). This paper contends that studying public opinion on social media can supplement – and add nuance to – information gleaned from survey data alone.

There is debate among scholars regarding the ability of protest to make pervasive, lasting shifts in public opinion. Recent research suggests that protest activity can shape political attitudes beyond the life of social movements; peaceful collective action events by African-Americans have been shown to garner support among whites by priming identities beyond race – like being American (*Mazumder*, 2018). Indeed, data shows that BLM protests in the summer of 2020 successfully shifted public attention toward anti-racist issues (*Dunivin et al.*, 2022). This research fails to account for variation in attitudes toward the movement – not all attention is positive attention. Indeed, another line of research suggests BLM protest activity only changed perceptions among low prejudice and politically liberal Americans. Any public opinion shift among conservatives following BLM protest activity was ”small and ephemeral” (*Reny and Newman*, 2021, 1499). As social media data can be scraped over time, this enables movement scholars to dynamically capture both protest and *reactions* to protest activity.

Social media data makes it possible to not only capture attitudes toward social movements, but also the motivations *behind* those attitudes. For instance, oppositional responses to #BlackLivesMatter are identifiable on Twitter because they often include instances of racism, bigotry, and hate (*Yang*, 2016). Social media sites provide digital spaces where political content, expressed via “personal action frames,” enable users to offer personal motivations for protesting the status quo (*Bennett and Segerberg*, 2015). Users can actively shape discussion, instead of just accepting frames from traditional news media. This has given people, especially youths of color, a platform to contest mainstream, color-blind ideology that dominates public discourse and marginalizes Black voices (*Cohen and Kahne*, 2011; *Carney*, 2016). Measuring attitudes toward social movements helps scholars study the effectiveness of protest activity in changing public opinion.

Studying movements online enables researchers to capture their evolution in a more nuanced way. The Black Lives Matter movement uses social media to spread information, raise awareness, and engage public discourse; social media platforms are where coalitions are built and meaning is collectively cultivated (*Bonilla and Rosa*, 2015; *Mundt et al.*, 2018). As a platform, Twitter enables a broad audience ”to alter and manipulate the movement’s construction of meaning” because it is both open

and participatory (*Ince et al.*, 2017, p. 1827). Counter hashtags like #AllLivesMatter arose as BLM gained prominence, an attempt to reframe conversations of police violence outside of a racial context; indeed, support for All Lives Matter is correlated with implicit racism, color-blind ideology, and narrow understandings of discrimination (*Ince et al.*, 2017; *West et al.*, 2021). As such, the two competing hashtags represent arguments that are "socially constructed, historically situated, and constantly changing" (*Carney*, 2016, 15). The constantly changing nature of discourse is precisely what makes studying social movements online so difficult.

Limitations & Difficulties in BLM research For scholars interested in the Black Lives Matter movement, tweets tagged with #BlackLivesMatter offer a potential window into the movement's protest strategy, tactics, and policy goals (*Freelon et al.*, 2016; *Ince et al.*, 2017; *Gallagher et al.*, 2018). The scale and scope of social media data is a double-edged sword. Capturing *attitudes* toward Black Lives Matter is difficult because the openness of social media makes movements – and public opinion towards movements – more dynamic. A hashtag does not signal support for a cause, but rather, engagement in a conversation. It would be inaccurate to assume that merely using the hashtag #AllLivesMatter automatically signals support for one ideology or another.

Black Lives Matter advocates use the #AllLivesMatter hashtag to engage color-blind ideology directly, keeping that hashtag from derailing the larger conversation about racial discrimination and police violence (*Carney*, 2016). The digital debate over hashtags – "do *Black* lives matter or do *all* lives matter?" – represents a wider debate over the role of racial frames in public discourse. Recent research into Black Lives Matter employs Twitter data, but focuses solely on sentiment of tweet text:

One limitation of this research is that the focus was on the overall sentiment of a tweet. Meaning, that although tone could be identified, the opinion conveyed in the tweet was not. In the future, it could be helpful to know which side of the issue a tweet fell as well as its overall sentiment so that there can be a better understanding of the public's opinion regarding BLM. (*Patnaude et al.*, 2021, 82).

In this context, a tweet's sentiment could relate to either police violence *or* to the social movement *opposing* police violence – overall political attitude is left ambiguous. Political discourse online is exceedingly complex and nuanced. The sample tweet in Figure 4.1 shows how the interplay between text and image manifests on social media.



Figure 4.1: THIS is how you fight for justice. #Ferguson #PoliceBrutality #BlackLivesMatter <http://t.co/FPIsVeRExR> @JNicoleBK

The text reads: "THIS is how you fight for justice. #Ferguson #PoliceBrutality #BlackLivesMatter <http://t.co/FPIsVeRExR> @JNicoleBK." Accompanying the text is a nocturnal photo of a protester throwing what looks like a smoke bomb. Ideally, automated content analysis would be able process both text and image features, as both modes of content are necessary to understand the attitude being expressed.

Social movements can change the frame of the conversation online to match the moment. The openness of social media creates fluidity in the collective creation of meaning (*Mundt et al.*, 2018). This paper the machine learning tools necessary for scholars to keep pace with the Even among BLM advocates, for instance, the purpose behind the #BlackLivesMatter hashtag is not static. For instance, in August 2014, mentions of Ferguson and police violence, paired with #BlackLivesMatter, were prominent on Twitter but decreased the rest of the year – while discussions of movement tactics increased (*Ince et al.*, 2017). This conflicts with the claim that the most shared BLM tweets were were not organizing protests, but rather, spreading news and information to increase BLM's visibility (*Freelon et al.*, 2016). Movement goals develop and change over time: grievances attract activist attention and as protests increase, so do discussions of *how* to protest. Discussions of police violence eventually give way to discussions of policy initiatives (*Ince et al.*, 2017).

The importance of images Visual content is vital to understanding contemporary social movements. Movements use images to increase attention and spread their message across social media (*Kharroub and Bas*, 2016). Visual frames evoke a range

of emotions that increase online mobilization and offline participation: enthusiasm, fear, sadness, and anger (*Marcus et al.*, 2000; *Williams et al.*, 2020). The Black Lives Matter has used images to raise visibility and increase awareness (*Freelon et al.*, 2016). Not only does an image increase online attention, it can increase diffusion; the presence of an accompanying image increases the likelihood a tweet will be retweeted (*Casas and Webb-Williams*, 2019). If BLM advocates share tweets to foster support for the movement – and they are using images to do so – then visual themes in the data should help identify political attitudes being expressed. Evidence suggests conservative news outlets are more likely than liberal outlets to use nocturnal, dark visuals of the BLM movement, often depicting protests as dangerous and violent (*Torres*, 2018). It follows that images, often sourced from news outlets, would help differentiate between tweets supporting and opposing BLM.

Multimodal deep learning models, trained on data from the social media site Weibo, have helped researchers detect violence in collective action events in China (*Zhang and Pan*, 2019). While this study does examine protest activity, it examines *attitudes* toward BLM, as opposed to characterizing movement protests as violent or non-violent. Multimodal deep learning models, trained on Twitter data, have been used to capture election incidents – long lines, broken machines, and other voter difficulties – that occurred during the 2016 U.S. Presidential Election (*Pineda*, 2022). The shortcomings of social media data is that no sample, however large, will be representative of a country’s population (*Zhang and Pan*, 2019). That said, social media data has already helped researchers observe and quantify political phenomena in real time. This work contributes to similar ends.

4.3 Collecting & Coding Black Lives Matter Tweets

This section reviews data collection and coding efforts. Data for this paper comes from a corpus of 40,815,975 tweets posted between June 1, 2014 and May 31, 2015 – originally collected for a comprehensive study on the origins and early development of the Black Lives Matter movement (*Freelon et al.*, 2016). The original study notes the power images in raising BLM’s visibility. Most images in the data depict some combination of police, victims, protesters, slogans, commentary, news media personnel, and/or celebrities: “The Black Lives Matter network is structured to distribute related content among and between news sites that are in a position to maximize and amplify visibility” (*Freelon et al.*, 2016, p. 17). This paper develops and validates tools that will enable researchers to study the massive amounts of social media

content generated by modern social movements.

To qualify for the original data set, a tweet must contain a keyword referencing a Black victim of police violence from between 2014 and 2015.¹ Tweets were downloaded in October 2019 and images were scraped in November and December of the same year. Tweets that had been deleted since the publication of the original study were excluded. The `twarc`² library in Python was used to "hydrate" the replication data. `Twarc`'s `hydrate` command reads a file of tweet identifiers as input and produces, as output, the corresponding metadata from Twitter's lookup API (including the tweet text and image URL). This process yielded 26 of the original 40 million tweets. Difficulties arise in manually sorting each of the 26 millions tweets by their respective attitudes: "Determining the proportional sizes of each category with an acceptable degree of precision would be a prohibitively difficult undertaking, and we did not attempt it here" (*Freelon et al.*, 2016, p.26).

This paper automates the analysis of text and image content, so future research can quantify social media data with "an acceptable degree of precision." Training and validation data consists of 1,900 tweets coded for attitude toward BLM – support or opposed (for the full coding scheme, see the next section). The majority of tweets are actually *retweets* with @-mentions, indicating dialogue between users. The inclusion or exclusion of retweets in training samples is a point of contention among social media scholars, but the authors of the original study include retweets and thus, this study does the same. As mentioned, retweets indicate a dialogue, and the conversational style of Twitter is one of its core features.

¹Keywords include: ferguson, michael brown, mike brown, eric harris, ezell ford, black lives matter, akai gurley, eric garner, kajieme powell, freddie gray, tanisha anderson, walter scott, victor white, tamir rice, jordan baker, tyree woodson, jerame reid, john crawford, yvette smith, tony robinson, phillip white, dante parker, mckenzie cochran

²<https://github.com/DocNow/twarc>

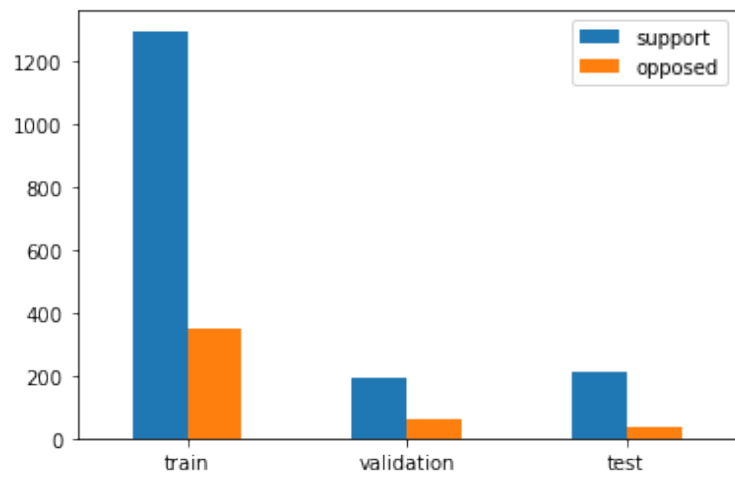


Figure 4.2: *Sample sizes and class proportions across the training, validation, and test data.*

Coding Scheme for Determining Attitudes Toward BLM

For each of the criteria listed below, the word "tweet" references *both* the text and image, not the text alone. That is, both modes of content were considered when assessing the attitude being expressed.³

Tweet is classified as "support" if it meets one or more of the following criteria:

Does the tweet explicitly support the Black Lives Matter movement?

Does the tweet express condolences to African-America victims of police violence or their families?

Does the tweet share news about Black Lives Matter, instances of police violence toward the African-America community, or protests against police violence?

Does the tweet offer criticism of institutions and their handling of police violence against the African-America community? Does the tweet portray the police in a negative light?

Does the tweet present the perspective of the African-American community, Black Lives Matter, and/or victims of police violence?

Does the tweet discuss BLM tactics and strategies, or encourage participation in protests?

Tweet is classified as "oppose" if it meets one or more of the following criteria:

Does the tweet offer criticism of protests, protest tactics, or BLM tactics more broadly?

Does the tweet offer criticism of the Black community? Specifically, does the tweet blame individual victims for their own deaths?

Does the tweet present institutional perspectives like those of the police, or the Justice Department?

Does the tweet portray BLM, victims of police violence, of the African-American community in a negative light?

³Tweets that did not feature images were excluded from analysis.



(a) THIS is how you fight for justice. #Ferguson #PoliceBrutality #BlackLivesMatter <http://t.co/FPIsVeRExR> @JNicoleBK



(b) This is not resisting arrest This is resisting DEATH. U say 11 times I can't breathe. What is wrong here. #EricGarner <http://t.co/YKdunc20sJ> @TyroneStevenson



(c) My blackness is not a weapon. #Ferguson #HandsUp #DontShoot @ErikaSlay

Figure 4.3: *Tweets supporting Black Lives Matter*

Support Figure 4.3 shows the text and images of tweets supporting the Black Lives Matter movement, taken from the training data. Tweets supporting Black Lives Matter often called for justice for Michael Brown and Eric Garner, and referenced police brutality explicitly, as in Figure 4.3(a). They would frequently combat narratives, propagated by opponents of BLM, that framed Michael Brown as a criminal and claimed Eric Garner was resisting arrest when he was killed, as in 4.3(b). Supporting tweets featured images of peaceful protests, candlelight vigils, and protest signs confronting systemic racism, as in 4.3(c). Often, hashtags like #HandsUp, #DontShoot, and #ICantBreathe clearly identify with a particular aspect of the Black Lives Matter struggle, whether it be a victim, an issue, or political stance.

Opposed Figure 4.4 shows the text and images of tweets opposing the Black Lives Matter movement. Tweets opposing the BLM movement often tried to re-frame the debate over police brutality, and systemic racism more broadly. To do this, opponents of the movement depict Michael Brown as a criminal and a gang member, as in 4.4(a). They often feature surveillance footage from the morning Michael Brown was shot, when he shoved a Ferguson Market clerk and stole a pack of cigarillos, as in 4.4(b). Additionally, opponents of the movement depict protesters rioting and looting; tweets in this category tend to criticize protests via sarcasm and memes, as in 4.4(c).

Training data was sampled specifically from November and December 2014, as these months were the periods of highest activity for Black Lives Matter. This timeline corresponds with the non-indictments of Michael Brown's and Eric Garner's killers (November 24 and December 3, respectively). The 1,900 coded tweets were randomly split into training ($n=1,650$) and validation data ($n=250$). To test how the model

BREAKING: Ferguson's Michael Brown PICTURED WITH A GUN – Flashing Gang Signs

Posted by Jim Hoff on Thursday, August 14, 2014, 11:09 PM

TRAIL LIFE...
Ferguson reason these photos haven't made the rounds...



This photo reportedly shows Michael Brown with a gun pointed at the camera.

(a) Live by the gun, die by the gun. #Ferguson #JusticeServed
<http://t.co/P82KeMCYGM>
NoQuarter_Given



(b) .owillis It's Michael Brown who needed to be monitored with a camera at all times. #Ferguson
<http://t.co/X66puWkRAW>
SamValley



(c) #GivingTuesday? Not for #MikeBrown supporters. For his #thug buds, it's #TakingTuesday! #Loot #Burn #Riot #shutitdown
<http://t.co/RBhSsacd6o>
@Monkey_Oil

Figure 4.4: *Tweets opposing Black Lives Matter*

would perform on out-of-sample data, an additional 250 tweets were coded from a corpus of tweets gathered during the 2020 iteration of the BLM movement, when protests erupted following the death of George Floyd. The bar graph in Figure 4.2 shows sample sizes and class proportions for each of the three subsets.

4.4 Multimodal Deep Learning for Tweet Classification

The use of both text and image data, called *multimodal* content analysis, has shown promise in deep learning applications in political science. This section provides an overview of deep learning models used throughout this study. Convolutional neural networks (CNN's) are layered structures for processing and classifying high-dimensional data (*Hastie et al.*, 2009). CNN's are considered *feed-forward networks* because data is passed forward from the input layer to the middle and output layers. An example of this process, taken from image layers of the combined text-image model, is shown in Figure 4.5. Training a CNN requires the use of weights that differentiate between relevant features in the input data; weights and input data are combined via matrix multiplication before being fed forward to the next layer. The network performs different operations on the data to capture important signals, reduce dimensionality, and find relationships in the data.

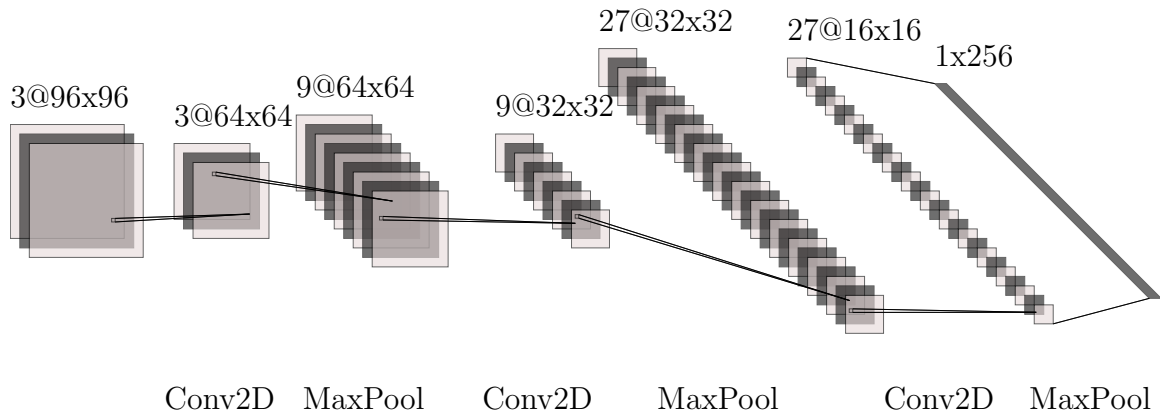


Figure 4.5: *Schematic of the image layers from the text-image model.* Convolutional filters create feature maps while pooling layers reduce the size of the data. This visual is specific to the image side of the architecture. The text side is parallel, with one-dimensional matrix operations instead of the two-dimensions required for images.

There are two phases for a CNN to process content: feature extraction and classification. During feature extraction, *convolutional* layers use filters that rotate across the data, searching for specific patterns and breaking down each individual input into *feature maps*. *Pooling* reduces the dimensionality of the data by either taking the average or the maximum among clusters of data points, reducing noise in favor of the strongest signals.

The full architecture of the combined text-image classifier is shown in Figure 4.6. Between the convolutional and pooling operations, a lot of effort is expended reducing images to their most essential features, making data easier to process in the final layers of the network. After feature extraction, the final layers of a CNN is concerned with classification. Feature maps are fed into the *dense* layers - where all of the inputs from one layer connect to all functions in the subsequent layer. This enables the network to detect non-linear relationships like image patterns or keywords that suggest a tweet’s membership in one class versus another.

4.4.1 Transfer Learning

Spurred by the costliness of training complex CNN’s, machine learning research has explored transfer learning – a technique where a model’s parameters, trained on a *source task*, is applied to a new *target task*. Transfer learning defies the assumption that training and target data must come from the same domain; often, researchers have a target task in one area of interest but use a model pre-trained on data from a separate, more general domain (*Pan and Yang, 2010*). This process works well with image data because images share a feature space. The lower levels of a neural network

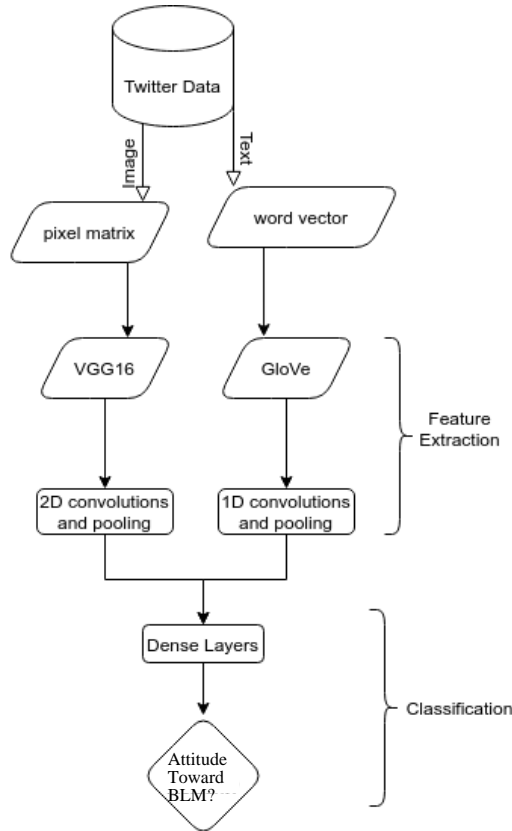


Figure 4.6: *The text-image model’s parallel architecture.* Content on both sides is represented in matrix form. A combination of transfer learning (from the VGG16 image classifier and GloVe word embeddings), convolutions, and pooling layers extract features from a tweet’s content. Once their features have been joined, dense layers learn patterns from both text and image features, before making a classification decision.

learn basic patterns, like edges and shapes, that are not exclusive to any particular domain, but rather, common to all pictures. Transfer learning seeks to improve performance by using the parameters of a pre-trained model to extract features from the target data. More formally:

Definition 3. Transfer Learning. *Given a source domain D_S and learning task T_S , a target domain D_T and learning task T_T , transfer learning aims to help improve the learning of target predictive function $f_T(\cdot)$ in D_T using the knowledge in D_S and T_S , where $D_S \neq D_T$, or $T_S \neq T_T$.*

Let domain D be defined by a *feature space*, X , and a marginal probability distribution, $P(X)$. For a domain of interest, $D = \{X, P(X)\}$, a task, T , has two parts: a set of labels Y and a predictive function $f(\cdot)$. A task simply refers to a function that predicts labels: $T = \{Y, f(\cdot)\}$. Much like research assistants, the predictive function

learns from examples. This training data consists of pre-labeled data $\{x_i, y_i\}$ where $x_i \in X$ and $y_i \in Y$. The ultimate goal is to estimate the parameters of $f(\cdot)$ that can predict the label $f(x)$, given a new instance x .

Assume there are only two domains of interest: a source domain D_S and a target domain D_T . Data sampled from the source domain data is denoted $D_S = \{(x_{S_1}, y_{S_1}), \dots, (x_{S_n}, y_{S_n})\}$ where $x_{S_i} \in X_S$ is a data point and $y_{S_i} \in Y_S$ is the corresponding label. Equivalently, the target domain data is denoted $D_T = \{(x_{T_1}, y_{T_1}), \dots, (x_{T_n}, y_{T_n})\}$ where $x_{T_i} \in X_T$ is the data point and $y_{T_i} \in Y_T$ is the corresponding label. Transfer learning is the process of applying knowledge gained from D_S to a task in D_T . This study uses parameters from three different pre-trained networks. The text-only classifier makes use of GloVe word embeddings, pre-trained parameters that measure word similarity via Euclidean distance (more below) (*Pennington et al., 2014*). As seen in Figure 4.6, the text-image classifier makes use of these embeddings on the text side and the VGG16 model on the image side - a very deep CNN whose frozen parameters do not update, but extract features from tweet images.

Parameter Tuning The overall number of layers, the size of the convolution filters, the type of activation function, and the type of pooling are all *tuning-parameters* - adjustable settings defined by the user that impact both the speed of training and the accuracy of the model. An enormous amount of time and energy is spent testing different combinations of tuning-parameters to find what configuration provides the best performance. The difficulty in parameter tuning is that there are no hard and fast rules about how to determine the optimal configuration; no two networks are built the same, as configurations are specific to the problem statement and classification scheme. There are certain heuristics that can guide experimentation (for instance, it is generally best to increase depth depending on the complexity of the image) but this requires trial and error.

Parameter tuning is a balance between potential improvements and costs to model performance. Increasing the number of filters helps improve accuracy, but this requires more training data. Increasing the size of the learning rate helps the optimization algorithm converge faster, but runs the risk of unstable training and sub-optimal parameters. A learning rate that is too small will result in a long training process that gets stuck at a local minimum. Too many filters in a model runs the risk of overfitting - when a model adjusts parameters to accurately classify training data without detecting the relevant patterns that let it classify unseen data.

Backpropagation During training, optimal weights are found via a process called *backpropagation*. This process occurs in three stages – the forward pass, the backward pass, and the gradient update. During the forward pass, the linear combination of weights and input data is computed, then fed forward layer-to-layer until the model outputs the predicted label. During the backward pass, the distance between the predicted and true labels is calculated using the *loss function*. The derivative of the loss function with respect to each weight, the *gradient*, is computed layer-by-layer starting from the output layer and going backward. The third step in backpropagation uses the calculated gradient to adjust the weights toward the steepest decrease of the loss function. This process of iteratively computing the gradient of the loss function with respect to each weight and then adjusting weights to minimize loss, called gradient descent, efficiently decreases error at each training step.

Benchmarks for Training Training performance is measured using three benchmarks: accuracy, cross-entropy loss, and F_1 score. Accuracy, defined formally in Equation 4.1 below, captures the percentage of tweets the model correctly classifies. We define y_i as the true value for data point i and \hat{y}_i as the model’s predicted label. The higher the accuracy, the better the model.

$$\text{accuracy}(y, \hat{y}) = \frac{1}{n_s} \sum_{i=0}^{n_s-1} \mathbf{1}(\hat{y}_i = y_i) \quad (4.1)$$

Categorical cross-entropy loss, defined in Equation 4.2 below, is calculated as a sum of separate loss for each class label per observation. As probability p diverges from the true label, cross-entropy increases – loss closer to 1 indicates poor performance, while a perfectly performing model would have a loss of 0.

$$\text{loss}(y, p) = - \sum_{c=1}^M y_{i,c} \log p_{i,c} \quad (4.2)$$

In the above equation, we define M as the number of classes, p as the predicted probability, and $y_{i,c}$ is a binary indicator (0 or 1) that indicates if c is the correct class. The algorithm uses the loss function to capture prediction error. Gradient descent then updates weights to minimize loss, thereby reducing error before attempting to classify another batch of data. Training neural networks is thus framed as an optimization problem: update weights to reduce error until you arrive at a global minimum.

$$F_1 = \frac{2}{\text{recall}^{-1} + \text{precision}^{-1}} = 2 \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}} = \frac{2\text{tp}}{2\text{tp} + \text{fp} + \text{fn}} \quad (4.3)$$

Equation 4.3 above shows how F_1 is computed as the harmonic mean of precision and recall. Precision, conceptually, is the positive prediction rate and recall is a measure of sensitivity. The above equation refers to true positives ("tp"), false positives ("fp"), and false negatives ("fn"). Combining them as F_1 gives a single measure of the model's ability to predict unseen data (the statistic is between 0 and 1, the higher the score the better overall performance).

GloVe word embeddings and text processing Originally introduced by *Pennington et al.* (2014) GloVe is an unsupervised learning algorithm for representing words as vectors in space. Word embeddings are a natural language processing technique where words are cast into a geometric space such that distances between words capture their semantic similarity. The closer the words in space, the more similar they are to one another. Representing words as these global vectors means representing words by their semantic neighbors in geometric space. The location of that word in the space is referred to as its *embedding*. The algorithm learns global word-word co-occurrence statistics from Wikipedia, and the resulting vectors represent linear relations between words.

Image Processing To better understand how a CNN processes image data, think of an image as a three-dimensional matrix consisting of pixel values. Images are represented by their pixel values, from 1 to 256, along three color channels – red, green, and blue (RGB). Data is normalized by dividing all values by the range (255). This ensures that all of the input data exists on the [0,1] scale which enables the network to converge faster during training.

During training, the CNN performs a series of matrix transformations on the data, so there cannot be any variation in dimensionality of the inputs. The training algorithm expects all images to have a square, uniform size, so all images are resized to 96 x 96. For data augmentation, pre-processing images can include transformations to provide the model more variation from its training input. Training images can be randomly flipped, zoomed-in, or tilted along different axes. This provides the model with examples that are better representative of the feature space, improving classification accuracy of unseen data.

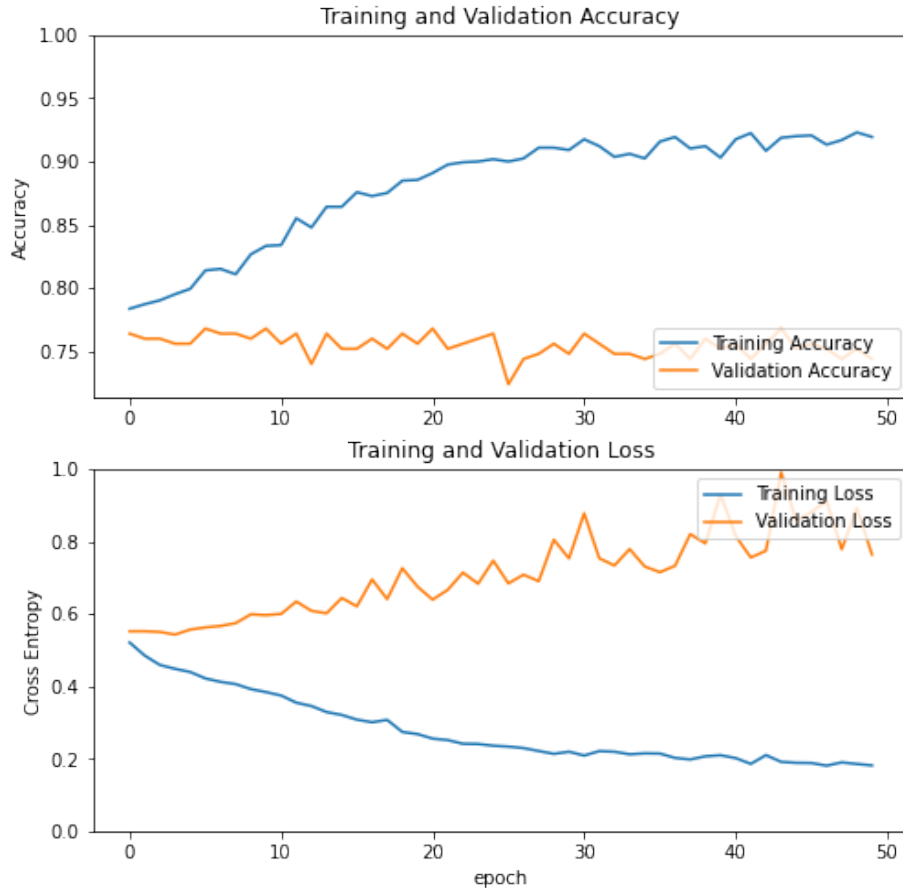


Figure 4.7: *Training metrics for the text-only model.* The model’s training and validation metrics immediately diverge. Even though the model shows high training accuracy, predictive performance on the unseen validation data steadily declines.

4.5 Results and Discussion

Figure 4.7 shows training and validation results for the text-only model. The model demonstrates high accuracy on training data, successfully classifying about 92% of tweets. Performance declines on validation set, however, suggesting that the model’s parameters are overfitting to the tweet text found in the training data. Indeed, training and validation metrics immediately diverge, suggesting that there is not enough information in the text alone for the model to accurately classify unseen data.

Class-specific results are shown in the confusion matrices in Figure 4.8. These illustrate why the model struggles on validation data. The top left and bottom right panels of the confusion matrices show what proportion of tweets that were accurately classified as opposed and support, respectively. The bottom left panel depicts the

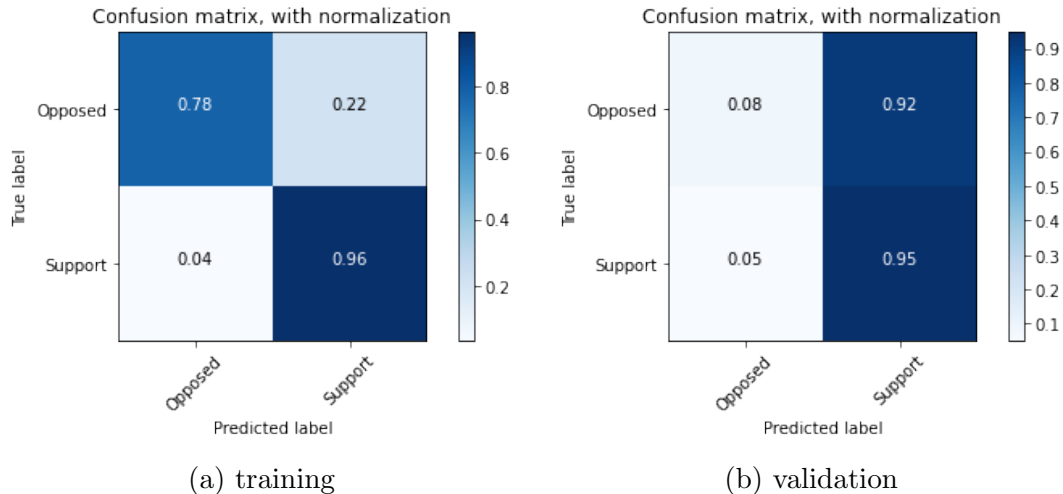


Figure 4.8: *Confusion matrices for the text only model.* The text only model suffers from overfitting, with a noticeable decrease in performance from training to validation. Note especially the upper left panel in (b). The text only model has trouble identifying attitudes Opposed to BLM in the validation data.

proportion of *opposition* tweets that the model *misclassified* as support. The top right panel depicts the proportion of *supporting* tweets that the model *misclassified* as opposed. These results indicate that the class imbalance in the training set was large enough to skew the model’s ability to identify opposition tweets, the minority class.

Looking at Figure 4.8, the model is able to accurately predict 96% of supporting tweets in the training data and 95% of supporting tweets in the validation data. The model has trouble identifying tweets opposed the Black Lives Matter movement. The model only identifies 78% of opposition tweets in the training data. This number declines to .08% in validation, with the model misclassifying 92% of opposition tweets as support.

Training and validation results for the combined text-image classifier are shown in Figure 4.9. The text-only and combined text-image classifier show comparable predictive performance on validation data (both models had an F_1 score of .67). The multimodal model achieves a similarly high training accuracy rate as the text-only classifier (about 92%). The key difference between the two models is performance on the minority class in the validation set. Although the text-image classifier similarly suffers from overfitting, this first 10 epochs show training and validation metrics moving together. This suggests the image data adds information that helps the neural network differentiate between the two classes.

Figure 4.10 shows the confusion matrices for the text-image classifier. The results



Figure 4.9: *Training and validation metrics for the text-image model.* Early results show parallel accuracy metrics for the training and validation data. The metrics eventually diverges, however, as the model’s parameters overfit to the training data.

indicate that the text-image classifier is better at predicting opposition tweets, as performance surpasses the text-only model on validation. Looking at Figure 4.11, we see similar struggles in the model’s attempt to classify out-of-sample tweets from the 2020 BLM protests.

Discussion

While it would be tempting to suggest “more data is necessary,” future research should strive for data whose features are more *identifiable*. Computational social science should continue developing tools for automated content analysis, as the streams of data with politically relevant content are only going to grow wider, their oceans more vast. To that end, social science training sets need to be more theoretically informed. Samples should be more attuned to the nuances that differentiate classes. Deep learning models show a high level of accuracy on training data, but predictive

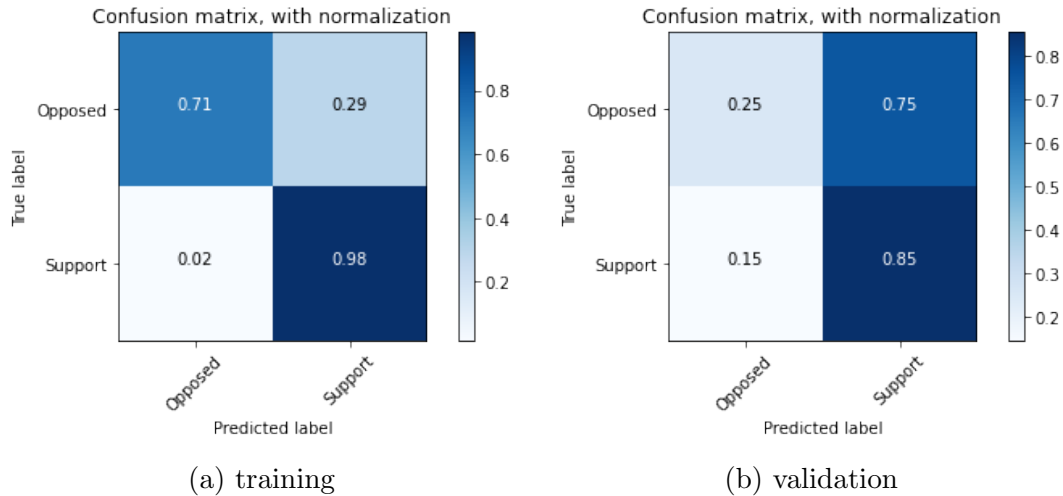


Figure 4.10: *Confusion matrices for the text-image model.* The model demonstrates high accuracy in identifying supporting tweets even on the validation set. The model does a decent job of identifying opposition tweets on the training data, but less so on validation set. Note that the text-image model does a better job of predicting opposition tweets in the validation set than the text-only model. This suggests that image features supplements information for minority classes.

power would benefit from balanced, well-defined classes.

One element from machine learning research that is missing from computational social science is the use of large-scale, standard training data. Training data in machine learning is standardized, but highly generic and void of context. Social science requires specialized data, specific to particular sub-fields or threads of research. Making data available from one study to the next, for the sake of replication, is not new; the purpose, however, must be adjusted slightly. If the stated goal of machine learning is to automate human tasks, this work suggests we are not far from that goal – both the text and text-image algorithms successfully replicated human coding efforts. The algorithms need improvement, however, when attempting to predict previously unseen data. That jump from learning to prediction will require continued aggregation, coding, and curation of politically rich content.

Avenues for Future Research Future research might examine how online public opinion to the Black Lives Matter movement fluctuates over time. Such work would require long term tweet collection via the Twitter Streaming API. The streaming API gives developers real-time access to approximately one percent of the 500 million tweets posted daily. The API requires a collection of terms to act as filters – only tweets that include one or more of the terms will be returned. Previous studies into

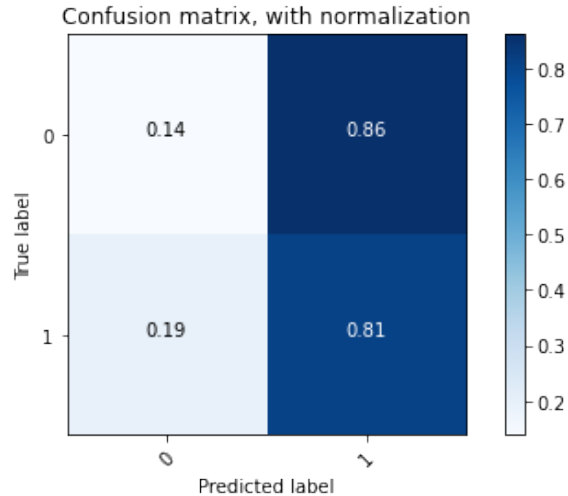


Figure 4.11: *Confusion matrix for the text-image model on the 2020 test data.* Data was collected in May and June 2020. Results indicate that, like the validation data, the model struggles to classify opposing tweets.

BLM have used as filters a combination of keywords relevant to the movement (“black lives matter”), the names of African-American victims of police violence (“Trayvon Martin”), and locations of prominent protests (“Ferguson”).

Another way forward in this vein would include collecting data on other websites and social media platforms – Reddit would certainly be of interest. Different subreddits are geared toward specific audiences, so a simple random sample might be problematic depending on the population of interest. One could stratify the sample by mining data from subreddits across the political spectrum. The other difficulty in this regard is the spontaneity of protest; even if the Black Lives Matter movement remains salient in American politics, its level of protest activity varies over time. Variation in activity, in some ways, would be helpful, as scholars can see how public opinion shifts in periods of high protest and if those shifts endure in periods of low protest.

Future research might examine how personal frames vary with attitudes toward Black Lives Matter. For instance, both supporting and opposing tweets reference justice – do varying attitudes signal varying conceptions of justice? Personal frames presented on social media are unique in that information is volunteered. Perspectives on protest activity are not primed, as they might be in a survey; public opinion on protests is offered unprompted. Future research might use the personal frames presented on social media to motivate survey questions. One approach might have respondents look at tweets directly – with different tweets signaling different personal

frames – and have respondents answer to what extent they agree with the opinions expressed. A potential treatment variable would be the presence of an image in the tweet, should research want to further explore the role of visual cues on attitudes; researchers could have the same frame presented with and without an image to see if there's an effect on the data, other variables held constant.

Social movement scholars could move beyond BLM to see how these methods capture attitudes related to other online movements. Black Lives Matter is hardly unique in its use of social media to raise awareness, organize, and spread information; the Arab Spring, #MeToo, and Occupy Wall Street are other examples of contemporary movements with a strong online presence. Capturing real-time response to these movements would be useful, especially as a short-term measure of the public's response to protest activity. This could be compared to survey data taken after protest activity declines in the intermediate- and long-term. Such analysis would provide a more complete understanding of public response to social movements. Further, the personal frames expressed online can motivate the questions regarding respondent's point of view. Beyond that, I imagine future scholarship will depend on the scholar.

4.6 Appendix I: Code Overview

```
#!/usr/bin/env python
# coding: utf-8

#####
#### I. DATA PRE-PROCESSING ####
#####

# Pseudo-code for Text and Image Tweet Classifier:
    Attitudes Toward Black Lives
    Matter
    DOI: 10.13140/RG.2.2.10898.53444

# For full code, visit:
https://github.com/apineda91/DeepLearn4PolComm
python libraries required: tensorflow, numpy, sklearn,
    nltk,
pandas, keras (part of tensorflow), matplotlib, os, sys

# Define function to plot_confusion_matrix
def plot_confusion_matrix

# Define columns and load data
cols = ['tweet_id', 'date', 'text', 'support', 'hashtags',
    ,
users', 'urls', 'media_urls', 'nfollowers', 'nfriends', 'file_name', 'path']

DATA = pd.read_csv('blm_apsa_sample3.csv', names = cols,
    dtype
pe = {'tweet_id':str, 'text':str, 'support':str, 'path':
    str,
'file_name':str})

# Check data types
```

```

# Define 'support' as binary variable

conditions = [
    (DATA['support'] == '0'),
    (DATA['support'] != '0')
]

values = ['0', '1']

DATA['support2'] = np.select(conditions, values)

# Spot check (always get a visual on your data to make
    sure
it looks like what you think it looks like)

# Drop null values
# Define path to images
# Check that the images
    # 1) exist
    # 2) are where you think they are

DATA = DATA[[os.path.isfile(i) for i in DATA['path']]]

# Shuffle data

DATA = DATA.sample(frac=1, random_state=1234).reset_index(
    dr
op=True)

# Test data is set aside until parameter tuning is done

test_data = DATA.sample(n=250, replace=False, weights=None
    ,
random_state=1234, axis=None)
DATA = DATA.drop(test_data.index)
len(test_data)

```

```

# Spot check
test_data.groupby('support').count()

# Validation data used to check for overfitting

val_data = DATA.sample(n=250, replace=False, weights=None,
    r
    andom_state=1234, axis=None)
train_data = DATA.drop(val_data.index)
len(val_data)

# Spot check
val_data.groupby('support2').count()
train_data.groupby('support2').count()
os.chdir('/media/alex/easystore/ids_monthly/BLM_images/
    apsa_
    sample3')

# Define data generators that can handle multi-modal
    inputs

def text_generator(a,labs, n):
    while True:
        for i in range(a.shape[0] // n):
            d2 = a[n*i:n*(i+1)]
            y_text = labs[n*i:n*(i+1)]
            yield d2, y_text

def multi_input_generator(df, x_image, y_image, x_txt,
    y_txt
    , b_size):
    t1 = input_imgen.flow_from_dataframe(
        dataframe=df,
        directory = './apsa_sample3',
        x_col = x_image,

```

```

        y_col = y_image,
        target_size=(img_width, img_height),
        batch_size=b_size,
        class_mode='categorical',
        validate_filenames=False)

t2 = text_generator(a=x_txt, labs = y_txt, n=b_size)

while True:
    d1,y = t1.next()
    d1 = np.expand_dims(d1, axis = 0)
    d2, y_text = t2.__next__()
    yield [d2, d1[0]], [y_text, y]

##### i. TEXT PRE-PROCESSING #####
### 1. Define hyper-parameters
### 2. Subset to only data and labels
### 3. Tokenizers for training text data
### 4. Tokenizers for val text data

##### Initialize the following data generators #####
### 1. train_generator
### 2. val_generator
### 3. test_generator

##### PREPARING GLOVE LAYER #####
### 1. build index mapping words in the embeddings set
to their embedding vector
### 2. compute and prepare embedding matrix
### 3. load pre-trained word embeddings into an Embedding
    la
yer

##### ii. IMAGE PRE-PROCESSING #####
### 1. load pre-trained CNN

```

```

### 2. check how many layers are in the base model
### 3. define which layers will be fine-tuned
### 4. freeze the layers you don't want trained

#####
##### II. MODEL CONSTRUCTION #####
#####

# TEXT-SIDE: EMBEDDINGS AND ADAPTATION LAYERS
sequence_input = Input(shape=(None, ), dtype='int64')
embedded_sequences = embedding_layer(sequence_input)
x_text = Conv1D(32, kernel_size = 5, activation="relu",
padding
ing = 'same')(embedded_sequences)
x_text = MaxPooling1D(5)(x_text)
x_text = Conv1D(32, kernel_size = 5, activation="relu",
padding
ing = 'same')(x_text)
x_text = GlobalMaxPooling1D()(x_text)
x_text = Dense(16, activation="relu")(x_text)
preds = Dense(2, activation='relu')(x_text)

# IMAGE SIDE: PRE-TRAINED CNN AND ADAPTATION LAYERS
x_image = image_model.output

x_image = Conv2D(4, 1, activation="relu", padding = 'same
')(
x_image)
x_image = MaxPooling2D(pool_size=(2, 2), strides=(1, 1),
padding
ing='same')(x_image)
x_image = Conv2D(4, 1, activation="relu", padding = 'same
')(
x_image)
x_image = MaxPooling2D(pool_size=(2, 2), strides=(1, 1),
padding

```



```

ding='same')(x_image)
x_image = Conv2D(4, 1, activation="relu", padding = 'same
')(
x_image)
x_image = MaxPooling2D(pool_size=(2, 2), strides=(1, 1),
pad
ding='same')(x_image)
x_image = Flatten()(x_image)
img_predictions = Dense(2, activation="relu")(x_image)

merged = Concatenate()([preds, img_predictions])

# ADAPTATION LAYERS
x = Dense(64, activation='relu')(merged)
x = Dense(32, activation='relu')(x)
x = Dense(16, activation='relu')(x)
x = Dense(8, activation='relu')(x)
x = Dense(4, activation='relu')(x)
#x = Dropout(0.5)(x)
main_output = Dense(2, activation='sigmoid', name = '
main_ou
tput')(x)
print("ROMA VICTOR!")

# MODEL TRAINING:
# 1. initialize a model with two inputs and one outputs
# 2. initialize optimizer
# 3. compile model
# 4. train model (save output in an object)

#####
##### III. EVALUATION AND VALIDATION #####
#####

# For confusion matrix, we need to compute predictions
with

```

```
our trained model
# 1. Grab training predictions
    # convert to an array of binary values
# 2. Grab validation and testing predictions
    # convert to arrays of binary values
# 3. Input predictions and true values into
    plot_confusion_m
atrix
```

CHAPTER V

Conclusion

This project used computer science methods to explore topics relevant to the academic study of *politics*. For various reasons, these concepts are difficult to quantify and thus, difficult to study. The preceding chapters introduced several innovative uses of multimodal deep learning to quantify political phenomena that appear online. As such, this project makes an important contribution to two sub-fields in political science: political methodology and political communication.

Current trends in political methodology advance automated content analysis. This is spurred by the availability of politically meaningful content found online. Whether from social media or news sites, parliamentary or court transcripts, this sudden supply of content has fueled the demand for methods capable of extracting meaningful insight from messy, unstructured data. This project, more than most, meets that demand by concisely traversing an enormous amount of concepts in political and computer science. The goal was to demonstrate deep learning’s ability to process multimodal content of interest to political scientists. While the focus was on text and image data, terabytes of politically significant content flow across the internet, daily, in the form of text, image, audio, and video.

This dissertation demonstrated deep learning’s capacity to capture nuance in text and image data. The level of granularity a model can capture depends on the level of granularity found in the training data. Findings indicate that deep learning algorithms are adept at identifying political phenomena in unstructured, messy data – like image and text content. Future research should take these methods and equip them with more nuanced, theoretically informed training data.

The first study examined racial politics in the context of *algorithmic bias*. This occurs when a machine learns prejudice somewhere during the training process. As machine learning applications grow ubiquitous in the real-world, so do the consequences of algorithmic bias. For instance, deep learning models are currently de-

ployed by surveillance systems used in the United States. This study tested under what conditions these types of models learn racial prejudice.

The second study examined election administration. Specifically, it captured anecdotes of voters having either a positive or negative voting experience at the polling place. When aggregated, these election incidents identify under what circumstances voters face difficulty in casting a ballot. This chapter captured election incidents during the 2016 U.S. presidential election reported on Twitter, finding that image data provides additional information important for the algorithm to identify minority classes.

The third and final study examined *political attitudes*. Specifically, this chapter captures attitudes toward the Black Lives Matter (BLM) movement. Capturing attitudes toward Black Lives Matter provides scholars with deeper insight into public opinion on racialized in the United States. Difficulty arises, however, because public opinion toward BLM – like the movement itself – is too dynamic for scholars to keep pace. This study captures political attitudes toward BLM during the 2014 and 2020 wave of protests.

Taken together, these chapters demonstrate both the promise and difficulties of applying machine learning methods to the academic study of politics. These tools, although powerful, are not as infallible as they seem in computer science literature. That area of research relies too heavily on abstract coding schemes and data divorced from the messiness of real-world definitions. Social scientists, at present, map directly from concept to coding scheme; this is an impatient approach. The correct mapping should be from concept to feature to coding scheme. That way, the coding scheme is more than just a list of examples, but rather, a coherent set of *characteristics*.

The project used computers to capture concepts relevant to the study of American politics. These concepts have been difficult for scholars to quantify, but the preceding chapters demonstrated an innovative method in political science. The project made use of *deep learning* to analyze *text and image data*. The substantive chapters demonstrated this method by examining components of algorithmic bias, elections, and protest activity.

The production of knowledge requires the synthesis of knowledge. I have done that – not as good as some but better than most. It is thus, with great humility and respect, that I ask my dissertation committee to grant me a Doctorate in Political Science and Scientific Computing.¹

¹Thank you and good night.

BIBLIOGRAPHY

BIBLIOGRAPHY

- Abilov, A., Y. Hua, H. Matatov, O. Amir, and M. Naaman (2021), Voter-fraud2020: a multi-modal dataset of election fraud claims on twitter, *arXiv preprint arXiv:2101.08210*.
- Bennett, W. L., and A. Segerberg (2015), The logic of connective action: Digital media and the personalization of contentious politics, in *Handbook of digital politics*, pp. 169–198, Edward Elgar Publishing.
- Bentele, K. G., and E. E. O’Brien (2013), Jim crow 2.0? why states consider and adopt restrictive voter access policies, *Perspectives on Politics*, 11(4), 1088–1116.
- Bobo, L. (1988), Attitudes toward the black political movement: Trends, meaning, and effects on racial policy preferences, *Social Psychology Quarterly*, pp. 287–302.
- Bonilla, Y., and J. Rosa (2015), # ferguson: Digital protest, hashtag ethnography, and the racial politics of social media in the united states, *American Ethnologist*, 42(1), 4–17.
- Boussalis, C., and T. G. Coan (2020), Facing the electorate: Computational approaches to the study of nonverbal communication and voter impression formation, *Political Communication*, pp. 1–23.
- Brady, H. E., and J. E. McNulty (2011), Turning out to vote: The costs of finding and getting to the polling place, *American Political Science Review*, pp. 115–134.
- Carney, N. (2016), All lives matter, but so does race: Black lives matter and the evolving role of social media, *Humanity & Society*, 40(2), 180–199.
- Casas, A., and N. Webb-Williams (2019), Images that matter: Online protests and the mobilizing role of pictures, *Political Research Quarterly*, 72(2), 360–375.
- Chen, M. K., K. Haggag, D. G. Pope, and R. Rohla (2019), Racial disparities in voting wait times: evidence from smartphone data, *Review of Economics and Statistics*, pp. 1–27.
- Cohen, C. J., and J. Kahne (2011), Participatory politics. new media and youth political action.

- Cottrell, D., M. C. Herron, and D. A. Smith (2020), Voting lines, equal treatment, and early voting check-in times in florida, *State Politics & Policy Quarterly*, p. 1532440020943884.
- Drakulich, K., K. H. Wozniak, J. Hagan, and D. Johnson (2021), Whose lives mattered? how white and black americans felt about black lives matter in 2016, *Law & Society Review*, 55(2), 227–251.
- Du, M., F. Yang, N. Zou, and X. Hu (2020), Fairness in deep learning: A computational perspective, *IEEE Intelligent Systems*.
- Dunivin, Z. O., H. Y. Yan, J. Ince, and F. Rojas (2022), Black lives matter protests shift public discourse, *Proceedings of the National Academy of Sciences*, 119(10), e2117320,119.
- Freelon, D., C. D. McIlwain, and M. Clark (2016), Beyond the hashtags:# ferguson,# blacklivesmatter, and the online struggle for offline justice, *Center for Media & Social Impact, American University, Forthcoming*.
- Gallagher, R. J., A. J. Reagan, C. M. Danforth, and P. S. Dodds (2018), Divergent discourse between protests and counter-protests:# blacklivesmatter and# alllivesmatter, *PloS one*, 13(4), e0195,644.
- Garcia, M. (2016), Racist in the machine: The disturbing implications of algorithmic bias, *World Policy Journal*, 33(4), 111–117.
- Grimmer, J., E. Hersh, M. Meredith, J. Mummolo, and C. Nall (2018), Obstacles to estimating voter id laws’ effect on turnout, *The Journal of Politics*, 80(3), 1045–1051.
- Hajnal, Z., N. Lajevardi, and L. Nielson (2017), Voter identification laws and the suppression of minority votes, *The Journal of Politics*, 79(2), 363–379.
- Hastie, T., R. Tibshirani, and J. Friedman (2009), *The elements of statistical learning: data mining, inference, and prediction*, Springer Science & Business Media.
- He, K., X. Zhang, S. Ren, and J. Sun (2016), Deep residual learning for image recognition, in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778.
- Herrnson, P. S., R. G. Niemi, M. J. Hanmer, B. B. Bederson, F. G. Conrad, and M. W. Traugott (2009), *Voting technology: The not-so-simple act of casting a ballot*, Brookings Institution Press.
- Herron, M. C., and D. A. Smith (2016), Precinct resources and voter wait times, *Electoral Studies*, 42, 249–263.
- Ince, J., F. Rojas, and C. A. Davis (2017), The social media response to black lives matter: how twitter users interact with black lives matter through hashtag use, *Ethnic and racial studies*, 40(11), 1814–1830.

- Islam, M. T., A. Fariha, A. Meliou, and B. Salimi (2022), Through the data management lens: Experimental analysis and evaluation of fair classification, in *Proceedings of the 2022 International Conference on Management of Data*, pp. 232–246.
- Jones, D., and B. Simons (2012), *Broken ballots: Will your vote count?*, CSLI Publications Stanford.
- Joo, J., and Z. C. Steinert-Threlkeld (2018), Image as data: Automated visual content analysis for political science, *arXiv preprint arXiv:1810.01544*.
- Karkkainen, K., and J. Joo (2021), Fairface: Face attribute dataset for balanced race, gender, and age for bias measurement and mitigation, in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pp. 1548–1558.
- Kharroub, T., and O. Bas (2016), Social media and protests: An examination of twitter images of the 2011 egyptian revolution, *New Media & Society*, 18(9), 1973–1992.
- King, B. A. (2020), Waiting to vote: the effect of administrative irregularities at polling locations and voter confidence, *Policy Studies*, 41(2-3), 230–248.
- Kordzadeh, N., and M. Ghasemaghahi (2022), Algorithmic bias: review, synthesis, and future research directions, *European Journal of Information Systems*, 31(3), 388–409.
- Lausen, M. (2008), *Design for democracy: Ballot and election design*, University of Chicago Press.
- Leach, C. W., and C. P. Teixeira (2022), Understanding sentiment toward “black lives matter”, *Social Issues and Policy Review*, 16(1), 3–32.
- Li, Q., S. Shah, M. Thomas, K. Anderson, X. Liu, A. Nourbakhsh, and R. Fang (2016), How much data do you need? twitter decahose data analysis, in *The 9th International conference on social computing, behavioral-cultural modeling & prediction and behavior representation in modeling and simulation*.
- Marcus, G. E., W. R. Neuman, and M. MacKuen (2000), *Affective intelligence and political judgment*, University of Chicago Press.
- Mazumder, S. (2018), The persistent effect of us civil rights protests on political attitudes, *American Journal of Political Science*, 62(4), 922–935.
- Mebane Jr, W. R., A. Pineda, L. Woods, J. Klaver, P. Wu, and B. Miller (2017), Using twitter to observe election incidents in the united states, in *Annual meeting of the midwest political science association, chicago*.
- Mebane Jr, W. R., P. Wu, L. Woods, J. Klaver, A. Pineda, and B. Miller (2018), Observing election incidents in the united states via twitter: Does who observes matter?

- Mozur, P. (2019), One month, 500,000 face scans: How china is using a.i. to profile a minority, *The New York Times*.
- Mundt, M., K. Ross, and C. M. Burnett (2018), Scaling social movements through social media: The case of black lives matter, *Social Media+ Society*, 4(4), 2056305118807,911.
- Pan, S. J., and Q. Yang (2010), A survey on transfer learning, *IEEE Transactions on knowledge and data engineering*, 22(10), 1345–1359.
- Patnaude, L., C. V. Lomakina, A. Patel, G. Bizel, et al. (2021), Public emotional response on the black lives matter movement in the summer of 2020 as analyzed through twitter, *International Journal of Marketing Studies*, 13(1), 1–69.
- Peng, Y. (2020), What makes politicians’ instagram posts popular? analyzing social media strategies of candidates and office holders with computer vision, *The International Journal of Press/Politics*, p. 1940161220964769.
- Pennington, J., R. Socher, and C. D. Manning (2014), Glove: Global vectors for word representation, in *Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1532–1543.
- Pettigrew, S. (2020), The downstream consequences of long waits: How lines at the precinct depress future turnout, *Electoral studies*, p. 102188.
- Pineda, A. (2022), Deep learning models for detecting election administration issues on twitter, *Unpublished manuscript*.
- Reyn, T. T., and B. J. Newman (2021), The opinion-mobilizing effect of social protest against police violence: Evidence from the 2020 george floyd protests, *American Political Science Review*, 115(4), 1499–1507.
- Ricanek, K., and T. Tesafaye (2006), Morph: A longitudinal image database of normal adult age-progression, in *7th International Conference on Automatic Face and Gesture Recognition (FGR06)*, pp. 341–345, IEEE.
- Simonyan, K., and A. Zisserman (2014), Very deep convolutional networks for large-scale image recognition, *arXiv preprint arXiv:1409.1556*.
- Stein, R. M., et al. (2020), Waiting to vote in the 2016 presidential election: Evidence from a multi-county study, *Political Research Quarterly*, 73(2), 439–453.
- Stewart III, C., and S. Ansolabehere (2015), Waiting to vote, *Election Law Journal*, 14(1), 47–53.
- Torres, M. (2018), Give me the full picture: Using computer vision to understand visual frames and political communication, URL: <http://qssi.psu.edu/new-faces-papers-2018/torres-computer-vision-and-politicalcommunication>.

- Torres, M., and F. Cantú (2021), Learning to see: Convolutional neural networks for the analysis of social science data, *Political Analysis*, pp. 1–19.
- Torres, M., and F. Cantú (2022), Learning to see: Convolutional neural networks for the analysis of social science data, *Political Analysis*, 30(1), 113–131.
- Wang, Y., Y. Li, and J. Luo (2016), Deciphering the 2016 us presidential campaign in the twitter sphere: A comparison of the trumpists and clintonists., in *ICWSM*, pp. 723–726.
- West, K., K. Greenland, and C. van Laar (2021), Implicit racism, colour blindness, and narrow definitions of discrimination: Why some white people prefer ‘all lives matter’ to ‘black lives matter’, *British Journal of Social Psychology*, 60(4), 1136–1153.
- Williams, N. W., A. Casas, and J. D. Wilkerson (2020), *Images as data for social science research: An introduction to convolutional neural nets for image classification*, Cambridge University Press.
- Wu, P., and W. R. Mebane Jr (2020), Marmot: A deep learning framework for constructing multimodal representations for vision-and-language tasks, *Manuscript in preparation*. University of Michigan. <https://www.patrickywu.com/working-papers/marmot-wu.pdf>.
- Yang, G. (2016), Narrative agency in hashtag activism: The case of #blacklivesmatter, *Media and Communication*, 4(4), 13.
- Yu, J., X. Hao, H. Xie, and Y. Yu (2020), Fair face recognition using data balancing, enhancement and fusion, in *European Conference on Computer Vision*, pp. 492–505, Springer.
- Zhang, H., and J. Pan (2019), Casm: A deep-learning approach for identifying collective action events with text and image data from social media, *Sociological Methodology*, 49(1), 1–57.