Pixels as Data: Training a ConvNet to Detect Racial Cues in News Images

Alejandro Pineda

July 18, 2018

Abstract

Political communication research has established the importance of news images for creating visual frames, eliciting emotion, and grabbing readers' attention (Abraham and Appiah, 2006; Iyer and Oldmeadow, 2006; Dahmen, 2012). Methods for automated content analysis, however, focus exclusively on text while ignoring the visual cues that create additional, latent meanings. My paper addresses the issue of automated image analysis with specific focus on detecting racial bias in the media. I use transfer learning and a pre-trained neural network to classify news images based on their racial composition. Examining coverage of poverty from different news sources, I explore how images create associations between social issues and demographics. Replicating findings from Gilens (1996), I find prominent pairings between African Americans and poverty. I argue that transfer learning not only provides a technique to train image classifiers on limited training data, but also gives scholars deeper insight into how we derive meaning from images.

1 Introduction

News media today is delivered as a constant stream of headline-image pairings. Social media, blogs, and news sites bombard individuals with messages about the groups, issues, and events that require attention. Racial bias in news coverage is well documented by political scientists, as media frames pair immigration with Latino/as, poverty with African Americans, and crime with both minority groups (Gilens, 1996; Dixon and Linz, 2000; Valentino, Brader and Jardina, 2013). In some instances, the group cue is made *explicit* and a specific demographic is named in the text of the article. Political communication scholars using automated text analysis can capture this group cue with a dictionary method (search for the occurrence of "Latino" or "African American"). In other cases, the group cue is *implicit*; no specific demographics are referenced in the text, but instead are depicted alongside the article. For scholars using automated text analysis, this group cue is not easily captured on a large scale.

Images create visual frames that deliver latent meanings to readers. Studying these frames on a large scale presents a problem for scholars. Political communication scholars need image classifiers to match the scale of the text analysis being performed. Drawing on machine learning research, I demonstrate how political scientists can use convolutional neural networks (ConvNets) and transfer learning to efficiently train models on limited data. This process treats images as stacks of matrices populated with pixel values and relies on powerful computing to train millions of parameters.

This study makes two contributions: 1) I replicate theoretically motivated findings that African Americans are disproportionately represented as impoverished and 2) I demonstrate how scholars can train a machine to replicate this analysis on a large scale. This paper proceeds as follows: Section 2 reviews literature in political psychology and political communication; Section 3 reviews relevant theory on computer vision, race recognition, and transfer learning; Section 4 introduces basic ConvNet architectures before detailing how the race classifier was trained; and Section 5 reviews results of the training process and implications of the study.

2 Images as Visual Frames

Political psychology has established the importance of news images for creating visual frames, eliciting emotion, and grabbing readers' attention (Abraham and Appiah, 2006; Iyer and Oldmeadow, 2006; Dahmen, 2012). Visual stimuli affect emotional responses that ultimately shape readers' policy preferences (Iyer and Oldmeadow, 2006). This linkage between images, emotions, and policy preferences is consistent across issues including: Middle East conflict,

stem cell research, collective action, immigration, and poverty (Brantner, Lobinger and Wetzstein, 2011; Iyer and Oldmeadow, 2006; Dahmen, 2012; Corrigall-Brown and Wilkes, 2012; Brader, Valentino and Suhay, 2008; Gilens, 1996). This study focuses exclusively on images of the poor, specifically, the overrepresentation of African Americans in discussions of poverty.

RQ1: What demographics does the news media associate with poverty?

Traditionally, content analysis of news requires combing articles, coding for specific features – its topic, subject, or tone. Increasingly, large-scale studies employ automated text analysis to code articles *en masse* for sentiment, emotion, tone, and topic (Young and Soroka, 2012; Russell Neuman et al., 2014). For instance, Soroka et al. (2017) analyze $7,203$ stories and over 52 millions Tweets relevant to economic news stories. Russell Neuman et al. (2014) examine 29 issues across $13,362$ social media and $4,573$ traditional news sources. These studies focuses solely on text, while technological and resource constraints exclude images from automated analysis. This exclusion is problematic for inquiries into race and political communication, as group cues are triggered, in part, via image (Mendelberg, 2001; Abraham and Appiah, 2006).

Text *and* image are requisites for understanding group cues because images and coded language combine to prime racial considerations without mentioning race explicitly (Brader, Valentino and Suhay, 2008; Mendelberg, 2001). It is the pairing of text and image that creates latent meanings (Valentino, Hutchings and White, 2002). The discipline focuses too narrowly on textual frames while ignoring visual ones: "through implicit visual propositioning, extra-textual information only implied in photographs juxtaposed with the text will affect perception of issues specifically stated in news text" (Abraham and Appiah, 2006).

RQ2: What demographics does the news media associate with immigration?

Measuring such information is vital to efforts in political communication and racial politics, given that African Americans are disproportionately represented as the perpetrators of crime as opposed to its victims (Dixon and Linz, 2000). Coverage of immigration mentions Latino/as at higher rates than other groups (Valentino, Brader and Jardina, 2013). Such depictions have tangible implications for public opinion, as racial stereotypes mediate support for policies (Fox, 2004). For instance, news overwhelmingly depicts African America as the benefactors of welfare, causing citizens to overestimate the proportion of African Americans living in poverty (Gilens, 1996, 2009; Hancock, 2004).

The way journalists frame issues affects public opinion surrounding those issues (Kinder and Iyengar, 1987). When news coverage associates social problems with certain groups, these associations become ingrained in public consciousness. Elites then trigger group attitudes to shape political preferences toward the associated social ills (Valentino, Hutchings and White, 2002). In short, group-issue associations depicted in the news leads to group-based considerations during policy preference formation. Scholars must turn to machine learning techniques if they want to understand visual frames on a large scale.

3 Computer Vision and Racial Recognition

Computer vision is an interdisciplinary field that combines statistics, computer science, and machine learning to explore how computers can glean meaning from images and video. Beginning with Krizhevsky, Sutskever and Hinton (2012), research has explored *convolutional neural networks* (or ConvNets) for image analysis tasks including facial detection, object and pattern recognition, and video classification . These machine-learned models consist of millions of parameters, organized into distinct layers that feed data forward from layer to layer.

Neural networks go back several decades, but recent design innovations combined with powerful computing have led to the recent surge in interest. ConvNets are powerful, flexible, and scalable, making them ideal for analyzing high-dimensional data (e.g., images and video). This flexibility is both a blessing and a curse. ConvNet architectures can be configured in so many different ways that training these models is more art than science. Researchers benchmark their models using various visual recognition competitions, like the ImageNet Large-Scale Visual Recognition Challenge (ILSVRC).

Although scholarship in this field focuses on performance – efficiency and accuracy – over substantive applications, recent advances have sprung several subfields focused on specific applications, many of which are of interest to political scientists. For instance, facial recognition tasks focuses on detecting demographic information like age, race, and gender from a person's facial features (Won, Steinert-Threlkeld and Joo, 2017; Wang, Li and Luo, 2016; Singh, Hegde and Atrey, 2017; Wang, Guo and Kambhamettu, 2015; Wang et al., 2016; Wang, Feng and Luo, 2017). Of specific interest in this paper is the research on racial recognition.

RQ3: In classifying racial images, can a ConvNet outperform human coders in terms of accuracy and efficiency?

Fu, He and Hou (2014) provides an in-depth survey on the race recognition field, including the facial regions that are most important to classifiers and the different applications of this technology. Research has discovered several facial features that machines use to determine a person's race: eyes, eyebrows, nose, lips, skin color, and hairline. Going forward, this paper operates under the definition of race offered by Fu, He and Hou (2014):

"A group G_i is called racialized if its members can be agglomerated by either explicit or implicit patterns (physical appearance observations, such as skin tones, morphology, or feature vectors, etc.) to be appropriate evidence of ancestral links to a certain geographical region. Accordingly, the race classification/categorization issue can be expressed as an exploratory image analysis from a statistical pattern recognition perspective" (p. 2486).

This study offers a new application to racial recognition research – and computer vision more broadly – by analyzing images from news sources and drawing directly on political communication research. Additionally, I provide social scientists the theoretical and technical knowledge needed to train a ConvNet on limited training data. The resulting model, *X-pert Contender* (or *X-pert* for short) competes with the field's state-of-the-art in terms of training accuracy.

3.1 Transfer Learning

This section provides a background on transfer learning – its motivations, formal definitions, and implications for racial recognition tasks.

Spurred by the costliness of training complex models, machine learning research has explored transfer learning – a technique where models knowledge gained on a *source task* is applied to a new *target task*. Pan and Yang (2010) combat the assumption that training and target data must come from the same domain. Often, researchers have a task in one domain of interest but have training data in a separate domain. For instance, this study is interested in facial images, but leverages a model trained on cats, dogs, and boats (the ImageNet database). When done correctly, transfer learning helps researchers avoid costly data labeling efforts.

Pan and Yang (2010) define domain D in terms of 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 (also called "categories" or "classes") and a predictive function $f(\cdot)$. Thus, 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 $D_S = \{(x_{S_1}, y_{S_1}), ..., (x_{S_n}, y_{S_n})\}$ 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 $D_T = \{(x_{T_1}, y_{T_1}), ..., (x_{T_n}, y_{T_n})\}$ 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. 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 $T_S \neq T_T$.*

The source task for this study, T_S, is the classification of images from the ImageNet database. As noted earlier, subsets of this data are used in benchmark competitions for computer vision scholars. Thus, the source domain D_S consists of the competition's training data: 10 million labeled images depicting over $10,000$ object categories. The diverse set of categories include various dog breeds, vehicles, instruments, foods, and household items (Russakovsky et al., 2015).

Recent work in this field uses a pre-trained model, AlexNet, to examine profile pictures on Twitter (Wang, Li and Luo, 2016; Krizhevsky, Sutskever and Hinton, 2012). The study analyzes the the racial composition of Clinton and Trump followers during the 2016 election, finding that racial minorities were more likely to follow Clinton and white voters were more likely to follow Trump. The classifier achieves an accuracy of 95.6%, suggesting that photographs from ImageNet share a feature space with racial images.

RQ4: Does an image classifier – trained to differentiate between cats and dogs – contain knowledge relevant to the classification of race?

If our target domain D_T is the population of all facial images, then our target task, T_T, is the classification of faces into racial categories. Clearly, $T_S \neq T_T$ because the tasks are defined by different sets of labels ($Y_S \neq Y_T$). In this study, I argue that the random assortment of images from the ImageNet competition (the source domain) shares a feature space with racial faces (the target domain). All of this to say: a machine can learn to classify a person's race by first practicing on dogs and cats.

Pan and Yang (2010) identify three main issues for the use of transfer learning: (1) what to transfer (2) how to transfer and (3) when to transfer. "What to transfer" asks which part of the knowledge can be transferred across tasks. This study transfers parameters from Inception-v3 – a state-of-the-art model trained on the ImageNet dataset mentioned above

(Szegedy et al., 2014). This approach assumes that the source task and the target task share parameters: "The transferred knowledge is encoded into the shared parameters...Thus, by discovering the shared parameters...knowledge can be transferred across tasks" (Pan and Yang, 2010). By training X-pert on the features discovered by Inception, this study determines whether or not transfer learning is a viable approach for social scientists.

"How to transfer" refers to the learning algorithms that maximize performance across tasks. While this study offers insight into ConvNets, it does not experiment with different methods of transfer learning. A simple "brute force" approach is used. Inception receives the target images as input and feeds its output directly to X-pert – the racial classifier that is the main contribution of this study. Inception does the heavy lifting of breaking down images and X-pert analyzes the results.

Finally, there is the concern of "when to transfer." As noted earlier, transferring knowledge from Inception to X-pert assumes the two models share parameters. If this assumption holds, classification performance should improve over the course of training. Otherwise, the features that Inception deems relevant will be useless to the racial classifier, resulting in stagnation and poor predictive performance. This study contributes to transfer learning, racial recognition, and political communication by testing whether the ImageNet photographs share features with images of substantive interest to political scientists.

4 Data and Methods

This section uses an extended metaphor to introduce basic ConvNet architecture, explains how the Inception model helps X-pert process images, and details the data used during training.

4.1 A Concrete Metaphor for ConvNets

Think of a convolutional neural network as a school building: each floor is a layer and each student is a node. When students are grouped into classrooms, they work together as *filters* that slowly break images down into their core components. Classrooms on the lower floors learn basic shapes, lines, and edges, while higher levels leverage this knowledge to learn complex patterns.

Students do not study images in their entirety, but rather, one portion at a time (like a slide show). Images are generally too large for nodes to analyze at once, so they are cropped into separate, equally-sized squares. The size of the squares, called the *receptive*

field, determines how much data the filters see at a time.[1]

Figure 1: Early filters capture basic shapes (left panel). Later filters develop this knowledge, working together to learn more nuanced patterns (center). Eventually, filters recognize high-level features like Obama's ear (right).

As the slide show continues, students notice *features*: an oval, a curved line, a fuzzy patch, et cetera. Whenever students notice a designated feature – like the triangle in left panel of Figure 1 – they signal to the next floor that a relevant pattern has been spotted. This is called an *activation* and is the primary interaction between layers.

Once the signal is sent, students hold a piece of trace paper against the screen and hurriedly sketch the triangle. This tracing is a *feature map*, similar to those in Figure 1. Each classroom on the first floor – each filter in the first layer – is responsible for learning a different pattern. In addition to a triangle filter, there is an oval filter, a curvy filter, et cetera; later in the network, these filters combine to find more complex features.

If the first floor is a convolution layer, then the second floor is called a *pooling layer*. Pooling is a way to reduce the height and width dimensions of an image. Square filters rotate across the feature map – similar to the slide show from earlier – grabbing pixels with the highest values and discarding the rest. Figure 2 illustrates how this process reduces pixel count, making it easier for later layers to look for relevant patterns. Simply put: pooling discards irrelevant information.

Classrooms on the first floor deliver feature maps only to classrooms directly above their own. Note that students do not visit the any other classrooms on the second floor. This is an example of *local connectivity*: nodes in the early layers only send signals to nearby nodes in the next layer. It is more efficient for nodes to only communicate with their neighbors as opposed to visiting each node in the proceeding layer.

Though crude, this metaphor demonstrates how early layers of the network are responsible for feature detection (convolutions) and dimension reduction (pooling). Convolution

[1]A larger receptive field means fewer slides per image, increasing efficiency, but this requires nodes to digest more information per slide, leading to inaccuracies. This trade off between efficiency and accuracy is a theme in computer vision, and machine learning more broadly.

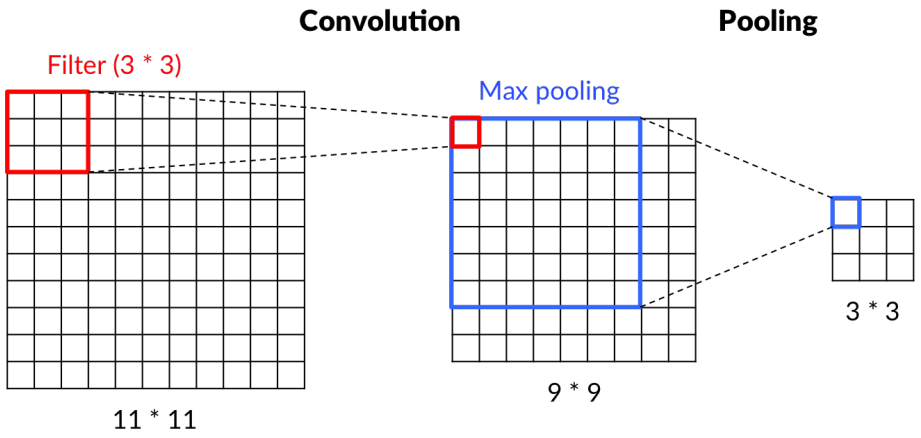


Figure 2: Convolutions detect patterns in the image, feeding them to pooling filters that drop low-strength pixels. This process filters out irrelevant parts of the image until its reduced to its most basic form; this makes it easier to detect the features that best determine categories (Figure source: https://saitoxu.io/2017/01/01/convolution-and-pooling.html).

layers learn features while pooling layers condense the resulting feature maps to a more manageable size.[2] This decreases computational cost, memory requirements, and the number of parameters that require training.

Another way ConvNets increase efficiency is by sharing parameters. Returning to our metaphor, imagine classrooms in the mid-levels combine sections, so our triangle class from earlier is joined by an oval class. Now, students can share their learned knowledge to learn more complex patterns, like the second panel in Figure 1.

By sharing parameters, different filters combine to detect increasingly complex patterns. This reduces the number of parameters that require training. Indeed, many techniques in ConvNets are aimed at the same goal: increase training efficiency by reducing the number of parameters that require training. Convolutions, pooling, parameter sharing, and local connectivity – all innovations specific to *convolutional* neural networks – are designed to make this process as efficient as possible.

Here, it is worth differentiating between ConvNets and their more general form, artificial neural networks (ANN's). ConvNets are a specific type of neural network designed to process images. ANN's refer to the computational model introduced by McCulloch and Pitts (1943), designed to mimic the way biological neurons communicate by firing signals. A key difference between the two is the way layers are connected. While the early layers of a ConvNet adhere to local connectivity, the last few layers resemble more traditional ANN's because they are

[2]Note that images are represented in three dimensions: height, width, and depth. As image gets smaller, the convolution layers stack feature maps together, increasing image depth.

fully connected. Figure 3 illustrates a basic ConvNet model where data is fed forward from layer to layer, mapping an image to a label.

Think of the last layers of a ConvNet as the top three floors of Haven Hall. Students on the fifth floor receive the feature maps from the classrooms directly below (local connectivity), study them, and then communicate with students on the sixth floor. Unlike previous floors, every student on the fifth communicates with *every student* on the sixth. The fifth and sixth floor are *fully connected*.

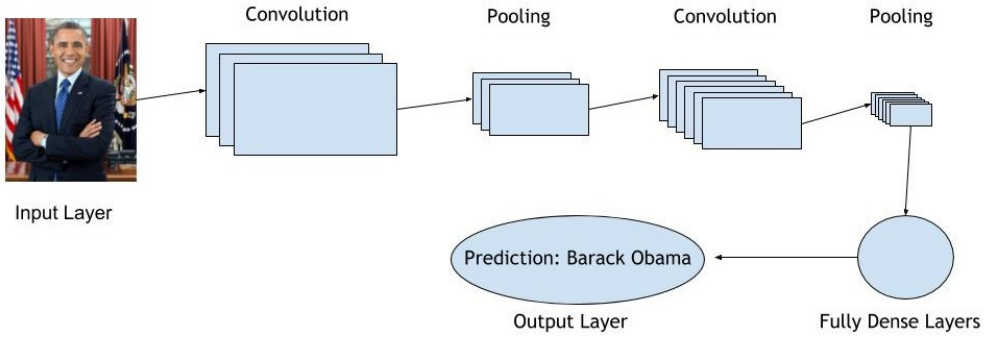


Figure 3: ConvNets feed data forward from layer to layer, breaking images down to their core features. The networks map images from feature space to class scores.

Instead of slide shows in separate rooms, local connectivity allows students to congregate in one large classroom. Feature maps are laid out on the floor, grouped according to their original image; images are clustered according to their category. Each student studies the features in a cluster and predicts the corresponding category. Predictions are sent to the seventh floor of Haven Hall – the *output layer* of the ConvNet – where graduate students tally predictions and categorize each image accordingly. The process just described – where nodes break down images into features, feed data forward, and make predictions – is called the *forward pass*.[3]

Now, Haven Hall is a specific building with its own *architecture*, but there is more than one way to design a building, just as there is more than one way to design a ConvNet. Research in computer vision seeks to find the right combination of convolution, pooling, and dense layers that best balance training efficiency and accuracy. Again, this is akin to Haven Hall, whose architect had to decide the number of floors in the building (layers in the network), the number of classrooms per floor (nodes per layer), and the overall layout of the building (the ordering of convolution, pooling, and dense layers).

[3]The next stage of training is called the *backward pass*. In this stage, tenured professors tell graduate students how wrong their predictions were. Graduate students then hunt down the undergraduates responsible for this humiliation, tell the little bastards how wrong *they* are, and then undergraduates work to be less wrong. For more information, see Section 3 of Appendix A.

In the machine learning literature, these different configurations are called *tuning parameters* and ConvNets are notorious for the seemingly infinite parameter space. The rest of this section details the transfer learning process and the data used for training/validation.

4.2 Transferring Knowledge from Inception to X-pert

The full, pre-trained Inception model was downloaded from GitHub. The model's layers were loaded into a Python dictionary, as shown in Figure 4. Each key in the dictionary corresponds to a specific layer, and the associated value detail the layer's purpose (convolution or pooling) and dimensions. The model's parameters are "frozen" so they will not do any additional learning. Inception acts as a large feature-extractor – a giant filter that we push images through. The filter takes images as input and produces feature maps as output, like those we see in Figure 5.

```
{'AuxLogits': <tf.Tensor 'InceptionV3/AuxLogits/SpatialSqueeze:0' shape=(?, 1001) dtype=float32
>,
 'Conv2d_1a_3x3': <tf.Tensor 'InceptionV3/InceptionV3/Conv2d_1a_3x3/Relu:0' shape=(?, 149, 149,
32) dtype=float32>,
 'Conv2d_2a_3x3': <tf.Tensor 'InceptionV3/InceptionV3/Conv2d_2a_3x3/Relu:0' shape=(?, 147, 147,
32) dtype=float32>,
 'Conv2d_2b_3x3': <tf.Tensor 'InceptionV3/InceptionV3/Conv2d_2b_3x3/Relu:0' shape=(?, 147, 147,
64) dtype=float32>,
 'Conv2d_3b_1x1': <tf.Tensor 'InceptionV3/InceptionV3/Conv2d_3b_1x1/Relu:0' shape=(?, 73, 73, 8
0) dtype=float32>,
 'Conv2d_4a_3x3': <tf.Tensor 'InceptionV3/InceptionV3/Conv2d_4a_3x3/Relu:0' shape=(?, 71, 71, 19
2) dtype=float32>,
 'Logits': <tf.Tensor 'InceptionV3/Logits/SpatialSqueeze:0' shape=(?, 1001) dtype=float32>,
 'MaxPool_3a_3x3': <tf.Tensor 'InceptionV3/InceptionV3/MaxPool_3a_3x3/MaxPool:0' shape=(?, 73, 7
3, 64) dtype=float32>,
 'MaxPool_5a_3x3': <tf.Tensor 'InceptionV3/InceptionV3/MaxPool_5a_3x3/MaxPool:0' shape=(?, 35, 3
5, 192) dtype=float32>,
 'Mixed_5b': <tf.Tensor 'InceptionV3/InceptionV3/Mixed_5b/concat:0' shape=(?, 35, 35, 256) dtype
=float32>,
 'Mixed_5c': <tf.Tensor 'InceptionV3/InceptionV3/Mixed_5c/concat:0' shape=(?, 35, 35, 288) dtype
=float32>,
 'Mixed_5d': <tf.Tensor 'InceptionV3/InceptionV3/Mixed_5d/concat:0' shape=(?, 35, 35, 288) dtype
=float32>,
 'Mixed_6a': <tf.Tensor 'InceptionV3/InceptionV3/Mixed_6a/concat:0' shape=(?, 17, 17, 768) dtype
=float32>,
 'Mixed_6b': <tf.Tensor 'InceptionV3/InceptionV3/Mixed_6b/concat:0' shape=(?, 17, 17, 768) dtype
=float32>,
 'Mixed_6c': <tf.Tensor 'InceptionV3/InceptionV3/Mixed_6c/concat:0' shape=(?, 17, 17, 768) dtype
=float32>,
 'Mixed_6d': <tf.Tensor 'InceptionV3/InceptionV3/Mixed_6d/concat:0' shape=(?, 17, 17, 768) dtype
=float32>,
 'Mixed_6e': <tf.Tensor 'InceptionV3/InceptionV3/Mixed_6e/concat:0' shape=(?, 17, 17, 768) dtype
=float32>,
 'Mixed_7a': <tf.Tensor 'InceptionV3/InceptionV3/Mixed_7a/concat:0' shape=(?, 8, 8, 1280) dtype=
float32>,
 'Mixed_7b': <tf.Tensor 'InceptionV3/InceptionV3/Mixed_7b/concat:0' shape=(?, 8, 8, 2048) dtype=
float32>,
 'Mixed_7c': <tf.Tensor 'InceptionV3/InceptionV3/Mixed_7c/concat:0' shape=(?, 8, 8, 2048) dtype=
float32>,
 'PreLogits': <tf.Tensor 'InceptionV3/Logits/Dropout_1b/cond/Merge:0' shape=(?, 1, 1, 2048) dtyp
e=float32>,
 'Predictions': <tf.Tensor 'InceptionV3/Predictions/Reshape_1:0' shape=(?, 1001) dtype=float32>}
```

Figure 4: The Inception model appears as a dictionary in Python with each key corresponding to a layer and each value providing details about the layer's purpose and shape.

The dictionary structure helps us navigate Inception's complex architecture, so that we can find the layers want to keep. Each entry corresponds to a layer, so for instance, you can

see several convolution layers early on 'Conv2d' followed by pooling layers 'MaxPool'. The 'Mixed' layers represent the Inception modules, a design innovation specific to this architecture (the details of which are beyond the scope of this paper). For this study, I kept the entire architecture and all parameters included, except for the output layer ('Predictions'). To transfer the model to its target task, I first had to reduce dimensions of the Prelogits layer, "squeezing" the output so that it can feed into the adaptation layers (imagine a giant water hose trying to connect to a much smaller one).

A two-layer, fully connected neural network was added on top of Inception (see Figure 6 for an illustration). The model was dubbed *X-pert Contender*.[4] The fully connected layer receives input from Inception-v3's PreLogits layer and outputs class scores for each racial category.

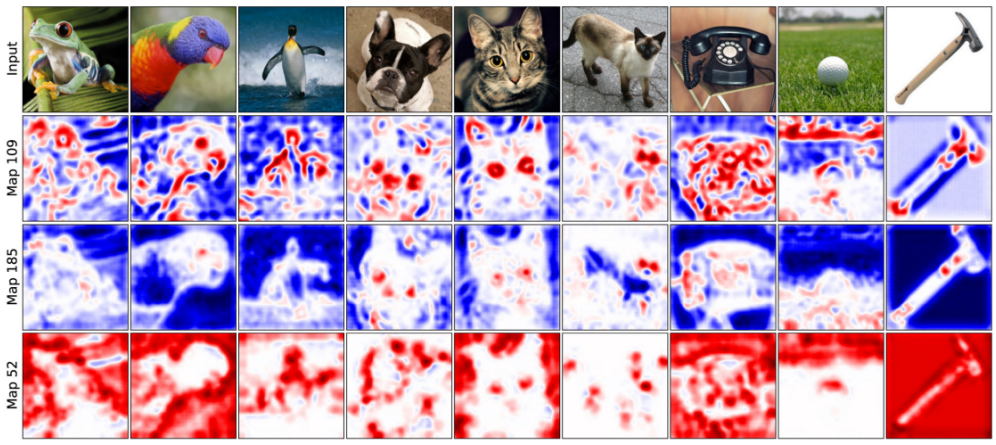


Figure 5: The features above are taken from the mid-levels of the Inception architecture. When applied to the MORPH data, Inception acts as feature extractor and X-pert learns the relationships between pixels that best determine race (Figure source: Zintgraf et al. (2017)).

X-pert actually refers to two different binary classifiers – BX-pert and LX-pert. BX-pert learned to differentiate between African Americans and Whites, while LX-pert learned to differentiate between Latino/as and Whites. The models were directed at two different sets of news images – one for poverty and one for immigration, detailed below.

4.3 MORPH Data

The training and internal validation sets for X-pert came from the MORPH database. This dataset contains 55,000 unique images of over 13,000 individuals from 2003 to 2007 and

[4]The name was conceived by a Wu-Tang Clan name generator. The author thanks Professor Christian Davenport for sending the link.

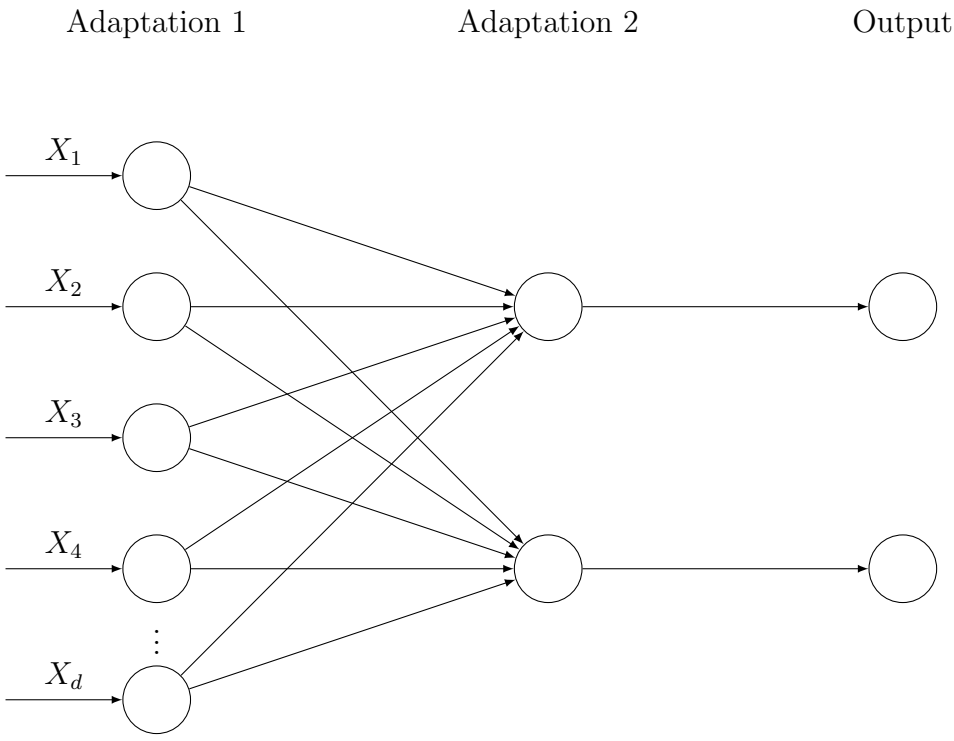


Figure 6: A diagram of X-pert, built on top of the much larger Inception-v3 architecture. The input data, X, is the output from Inception – feature maps of the original race images. X-pert actually refers to two different binary classifiers, BX-pert and LX-pert ("big" and "little", respectively). There models differ in two ways. BX-pert has 32 nodes in its first layer ($d = 32$ above) while LX-pert only has 15 nodes ($d = 15$). Additionally, BX-pert is trained to differentiate between African Americans and White people, while LX-pert is trained to differentiate between Latino/as and White people.

corresponding meta-data on gender, age, and race. The data was originally collected for research on aging, but has several desirable qualities our purposes: images have a standardized size, format, and naming convention. Metadata makes it easy to divide images into their racial categories. Subjects' faces are front and center (see Figure 7), making it easier for the machine to learn race-discriminant features (eyes, eyebrow, nose, mouth, and skin tone) without having to deal with extraneous objects in the image.[5]

The validation sets give us an idea of how the model performs in classifying unseen data. Variation in colors, filters, the prominence of faces in the photograph, and the presence of extraneous objects can all cause error. Machine learning practitioners use validation sets to identify problem cases that might arise during analysis of the target dataset. Ideally, validation sets (and training sets) resemble the type of data/images that the machine will ultimately analyze.

Figure 7: Samples images from the MORPH database. While data was originally used for research into aging, work by Wang, Li and Luo (2016) suggests that MORPH can be used in conjunction with transfer learning to build a racial face classifier.

Image Preprocessing

All training images from the MORPH dataset were randomly cropped so that faces appear off-center. The cropped images are resized to 299 x 299. For additional distortion, each image had a 50% probability of being flipped horizontally. Finally, colors were represented in 32-bit floating point format, ranging from 0.0 to 1.0.

Each class had an equal number of training instances from the MORPH dataset, ensuring no racial category was overrepresented ($10,000$ images per class for BX-pert and 1500 per class for LX-pert). Data was then shuffled before a portion was set aside for validation purposes.

[5]For more information on MORPH, see: http://people.uncw.edu/vetterr/MORPH-NonCommercial-Stats.pdf

Training proceeded as follows: the model received a batch of images from the training set, it made predictions based on extracted features, error was then calculated, and parameters were adjusted to reduce error. The model was fed the next batch and training continued. To improve accuracy, training data is generally split up into batches and iteratively fed into the classifier. A single epoch is when the machine sees all of the training data *once*. For instance, a training set of n=100 can be split up into 4 batches of 25. It would take the machine 4 iterations to complete a single epoch.

Table 1: Training and Testing Images

	Nodes	Training MORPH	Test MORPH	News Original	News Cropped	Epochs
BX-pert	34	16,000	4,000	113	173	20
LX-pert	17	2,700	300	138	185	10

Table 1, above, details the differences between the two models – their size, training data, validation data, and the number of epochs.

4.4 News Images of Poverty and Immigration

For this study, news images serve two purposes: 1) they provide data for external validation of X-pert and 2) they provide initial findings for the substantive questions related to race in the news.

For stories related to poverty, I visited the website of various news sites and performed a keyword search for articles related to "poverty" and "homelessness." I limited the search to articles in U.S. Domestic News, but placed no limit articles' publication date. For an image to be considered, the text of the article had to explicitly reference that the subject was either jobless, impoverished, reliant on Medicaid, homeless, or all of the above. This excluded several images of politicians, government workers, and advocates. While these image subjects may be of interest, they do not convey the poor and thus, do not have bearing on RQ1.

Gathering immigration images was more straightforward. I accessed the Associated Press Images Collection, courtesy of University of Michigan Library, and performed a keyword search for photos related to "immigration." I restricted my search to the "U.S. Domestic News" section and a publication date between 2000 and 2014 (to prevent from oversampling depictions of the current president). After sorting the results by relevance, I downloaded the first 136 images that met the criteria specified in Figure (coding scheme). Note that I did not look exclusively for images of immigrants, but rather, images that accompanied stories

on immigration *as an issue*. To determine if a story was relevant, I simply read the caption.

For inclusion into either dataset, images had to meet the criteria outlined by a coding scheme that filtered out faces the machine could not be reasonably asked to classify (codebook available upon request). Subjects of interest needed to face forward, their faces unobstructed by protest signs, microphones, or other people. Additionally, faces needed to be large/prominent enough that the model could reasonably assess race discriminating facial features – eyes, brow, nose, and lips were emphasized.

Figure 8: The left two images are from the poverty dataset and the right two are from the immigration dataset.

The resulting set of images was coded for the relevant race of each subject: Black-White for images of poverty, and Latino-White for stories related to immigration. Figure 8 shows a sample image from each category/issue. I determined race by examining the facial features deemed relevant by Fu, He and Hou (2014), discussed earlier. To further test the model's performance, each image was cropped so that only the face remained. While this was time-consuming, the cropped faces were meant to simulate bounding boxes used in computer vision research.[6] Validation tests were performed on both the original and cropped versions of the data.

Benchmarks for training

Training performance was measured using two benchmarks: accuracy and cross-entropy loss. Accuracy, defined formally in Equation 1 below, measures the percentage of images the model correctly classifies. We define y_i as the true value of subject i*'s* race, and $\hat{y}_i$ as the model's predicted label. The higher the accuracy, the better the model.

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

[6]An in-depth discussion of this technology is beyond the scope of this paper, but the intuition is thus: facial recognition technology places a bright box around each face in a photograph and the classifier focuses exclusively on whatever portion(s) of the image is bounded.

Cross-entropy loss, defined in Equation 2 below, measures performance of binary classifiers on a scale from 0 to 1. As predicted probability p diverges from the true label, cross-entropy increases; loss closer to 1 corresponds with poor performance, while a perfectly performing model would have a loss of 0.

$$\texttt{loss}(y, p) = -(y \log(p) + (1 - y) \log(1 - p)) \tag{2}$$

In above equation, we define p as the predicted probability and y as a binary indicator (0 or 1) if the model correctly predicts a subject's class.

5 Results and Discussion

This study asks, first, what demographics are associated with poverty in the news? Articles on poverty were found with a keyword search from various news organizations (CNN, Newsweek, New York Times, and the San Francisco Chronicle) and accompanying images were coded for the presence of African Americans or Whites. As Figure 9 indicates, the majority of poverty images (58.4%) were of African Americans. While this sample is relatively small ($n = 113$), it provides an initial estimate of how the media portrays the indignity of poverty.

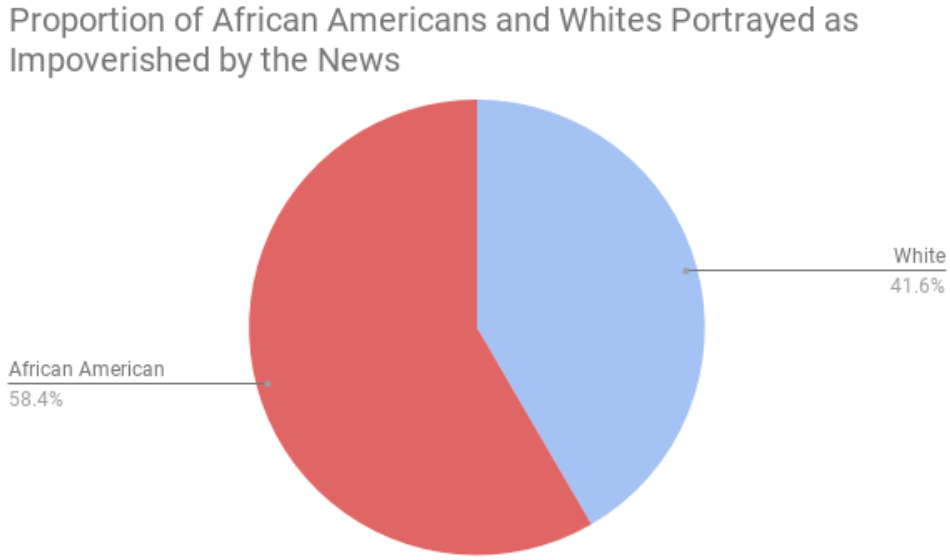


Figure 9: The percentage of poverty images represented by each race.

Second, this study explores what demographics the news media associates with immigra-

tion. Do images from the Associated Press depict more Latino/as or more Whites alongside coverage of immigration issues? Images were downloaded from the Associated Press Images Collection and coded for the presence of Latino/as or Whites. As Figure 10 indicates, the majority of images featured White people (60.7%). While this finding is counterintuitive, note that the vast majority of White photographs were of politicians debating immigration policy.

Further, the coding scheme used to select images required faces to prominent, large, and unobscured. The vast majority of images the query returned were of Latino/as; however, subjects were often in protest crowds, resulting in faces that were either too small, or obscured by protest signs, hats, sunglasses, and other people. Thus, we cannot draw definitive conclusions from this dataset on the media's portrayal of immigration without a larger sample.

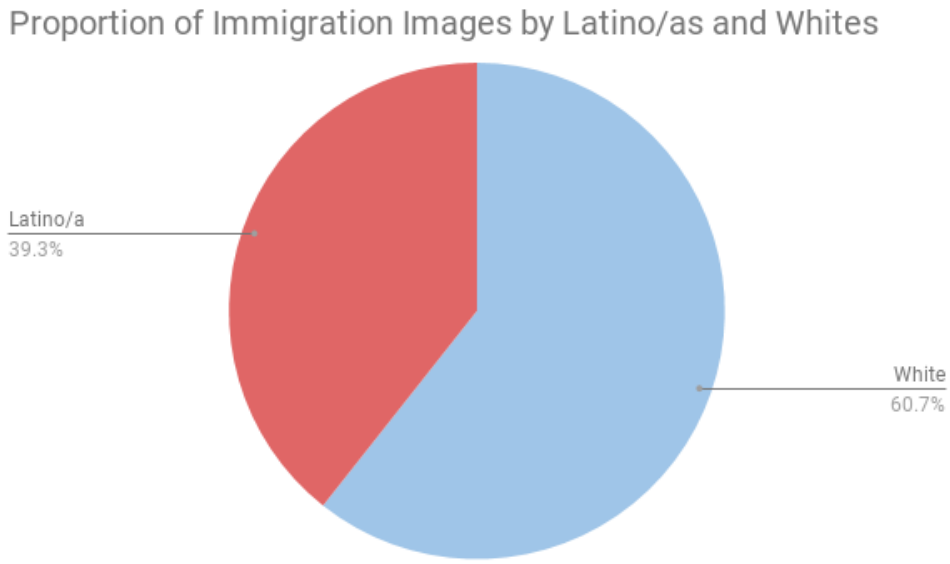


Figure 10: The percentage of immigration images represented by each race.

Third, this study examines whether a ConvNet can outperform human coders in the classification of racial images. In particular, do ConvNets offer a more accurate and efficient solution than the manual coding of images? Undeniably, the answer is yes. Table 2 shows the top training accuracy achieved by each version of X-pert, in addition to test accuracy across the three validation sets.[7] Note that both versions of X-pert approach human performance during training, but accuracy drops off for the validation sets, suggesting over fitting.

[7]As of this writing BX-pert is still analyzing the test data from MORPH. The model is running on Professor Mebane's server, which is bogged down by a nameless graduate student who is scraping every website ever to perform some sort of heathen text analysis.

That is, the nodes are learning features specific to the training data that do not appear in other settings. Machine learning has several techniques to fix this, including dropout and regularization. Both techniques will be explored in future studies.

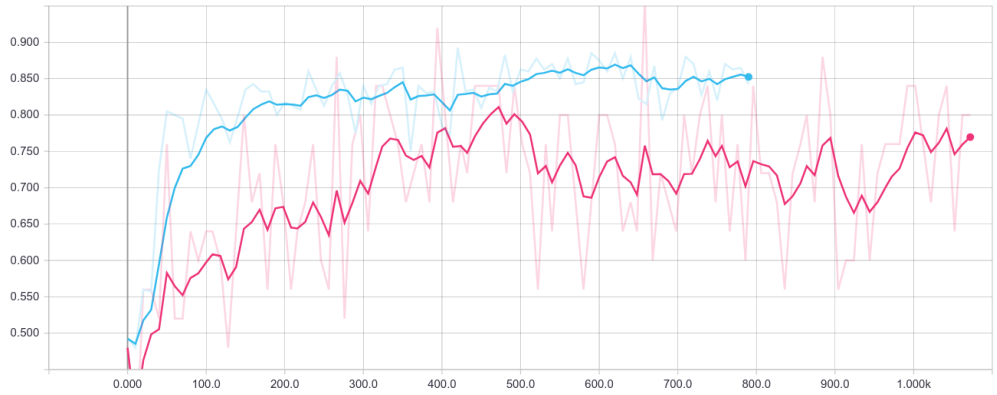


Figure 11: Training accuracy (y-axis) as a function of global steps (x-axis). BX-pert (the blue line, above) achieved a top training accuracy of .89. LX-pert (the pink line) achieved a top training accuracy of .96 after just 1 hour and 18 minutes of training.

Perhaps most striking is how well LX-pert performed despite limited training data ($n = 2,700$), few nodes (17), and a short amount of training time (10 epochs over 2 hours). State of the art models require hundreds of thousands of images, thousands of nodes, and *weeks* of training. All of this to say: early results indicate that X-pert has enormous potential.

Table 2: Accuracy and Efficiency Benchmarks Across Datasets

	Training	Test MORPH	News Original	News Cropped	Training Time
BX-pert	.89	-	.57	.68	17 hrs, 45 mins
LX-pert	.96	.74	.44	.62	2 hrs, 7 mins

Finally, this study asks: does a model trained to differentiate between cats and dogs contain knowledge relevant to race classification? A transfer-learned model, X-pert, learned to classify images of Blacks, Whites, and Latino/as with the help of Inception – a ConvNet trained on a completely unrelated dataset. The answer is yes: Inception's parameters helped X-pert learn to detect race. As Figure 11 indicates, X-pert showed steady improvement in accuracy over the course of training. Indeed, Figure 12 shows loss at a steady decline.

Machine learning theory states that if source and target domains *do not* share a feature space, transfer learning will not be successfuly and may actually hurt performance (Pan and Yang, 2010). Not only was performance not hindered, it steadily improved over the course of training. This suggests that the random assortment of images in the ImageNet database

share a feature space with the racial images.

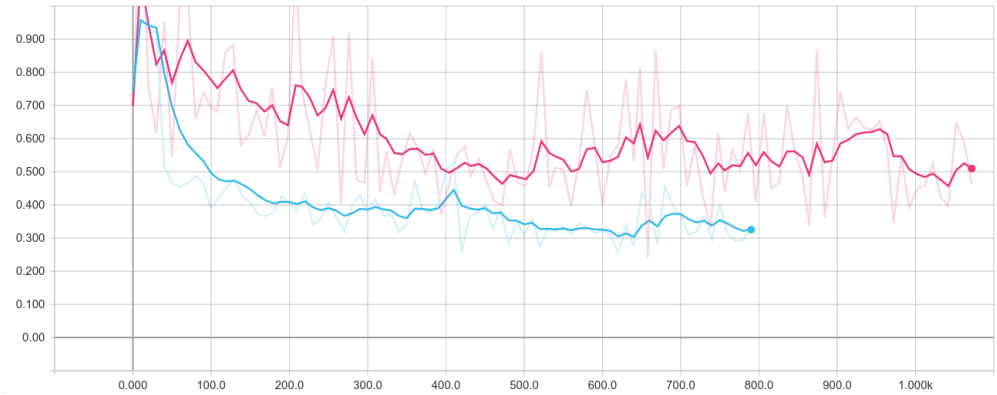


Figure 12: Loss (y-axis) as a function of global steps (x-axis). As above, BX-pert is presented by the blue line and LX-pert is represented in pink. Loss steadily decreases over the course of training, reaching a minimum of .29 and .25 for BX-pert and LX-pert, respectively.

Within social science, machine learning techniques should be applied to images in a similar vein as they are applied to text; that is, machine learning should not replace human coding, but rather, augment it. The development of a coding scheme and validation of machine's predictions are both still necessary in this paradigm (Grimmer and Stewart, 2013). Beyond a technology that helps researchers sort data, ConvNets offer insight into how we derive meaning from images. They highlight the components of a photograph that are essential to its definition.

Images have a hierarchical structure: any given image is composed of objects – a person or group of people, a fire hydrant, a cat – and those objects are composed of features. In seeking to understand how a machine makes sense of such noisy data, this hierarchical structure helps social scientists think about images, as unites of analysis, in a standardized way.

Appendix A Neural Networks

A.1 Single-Layer Networks

For data $X \in \mathbf{R}^d$, assume X is separable into classes $y \in \{1, 2, ..., k\}$ where assignment to class k means x_i shares relevant *features* with other data points in k. We need a statistical model that can learn the relevant features in X that map onto the associated classes in y:

$$X \mapsto y.$$

The model needs to understand the *relationships* between data points that *best determine class*. For instance, a simple linear model might look like:

$$\hat{y}(x, w) = \sum_{j=1}^{d} w_j x_j \tag{3}$$

where w are model parameters that capture relationships between x_i and $x_{i'}$ and $\hat{y}$ is the predicted classes as a function of x and w. For a binary classification task – only two classes – this model becomes:

$$\hat{y}(x, w) = \text{sign} \sum_{j=1}^{d} w_j x_j \tag{4}$$

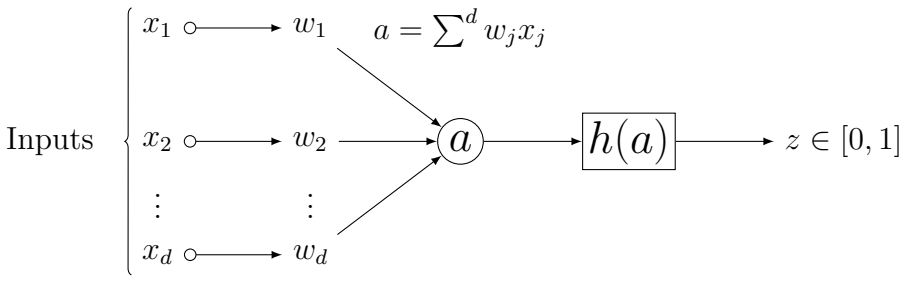
with classes defined as $[-1, 1]$ or even $[0, 1]$. For instance, social scientists working with a binary dependent variable should be familiar with logistic regression. We might use such a model for our purposes, of the form:

$$P(y = 1|\ x, w) = h\left(\sum_{j=1}^{d} w_j x_j \right) \tag{5}$$

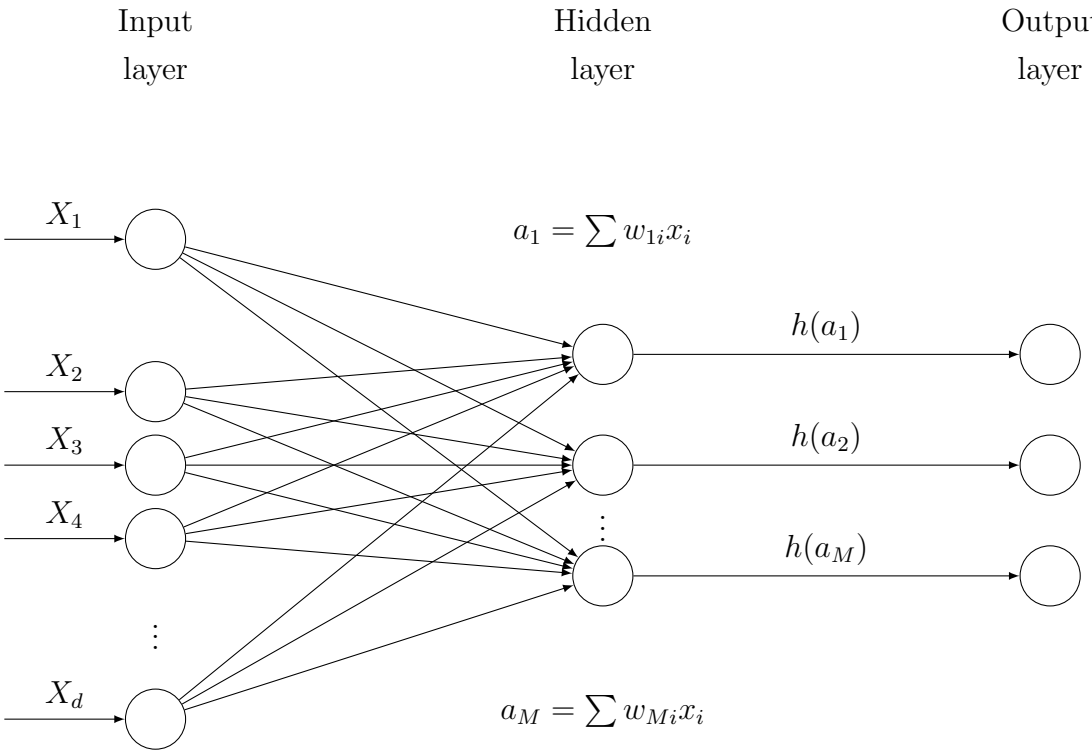
where h is the logistic function

$$h(a) = \frac{1}{1 + e^{-a}}.$$

With X in lower-dimensions, this model would be a reasonable approach, but let us assume that X has a large feature space and thus, requires a classifier that can scale well. To this end, we can represent our logistic regression graphically, as a network.



This particular form is called a *single-layer perceptron* and is the essential building block of neural networks. Note three essential components: an **input layer** (where X is fed into the model), **a node** (containing the dot product $X^T w$) and an **output layer** (re-labeled as z) that predicts class. Additionally, the logistic function used in Equation 3 is generalized to an activation function $h(a)$ that exists along the edge of the network (we will see that several different functions are used in this role). If we set $\hat{y} = a$ the single-layer perceptron approximates a logistic regression. Indeed, as we expand to more classes, we have more boundaries to consider, and we need a more powerful classifier.



We can capture more intricate relationships if we increased the number of weights in the model, as above. To handle the increase in weights, we add more nodes $\{a_1, a_2, ...a_M\}$ to

the model. By increasing the number of nodes in the "hidden layer" we can approximate any boundary or function (we call the inner layers of a network hidden because they do not go directly observed). Note that we maintain the same activation function, but it gets fed different values depending on what edge we are looking at. This structure can handle high dimensional data, but it can handle even more complexity by increasing the number of hidden layers in the network, as we see in the next section.

A.2 Multi-Layer Networks

A.2.1 Fully Connected (Hidden) Layers

If K is the number of classes, then Z_m represents the derived features, created from linear combinations of the inputs (in this case, pixels). Y_k is target measurement for class K. Our features, Z_m, are formally defined below:

$$Z_m = h(w_m^T X), \qquad m = 1, ..., M \tag{6}$$

These features get fed into the hidden layers (so called because they go unobserved):

$$T_k = \beta_{0k} + \beta_K^T Z, \qquad k = 1, ..., K$$

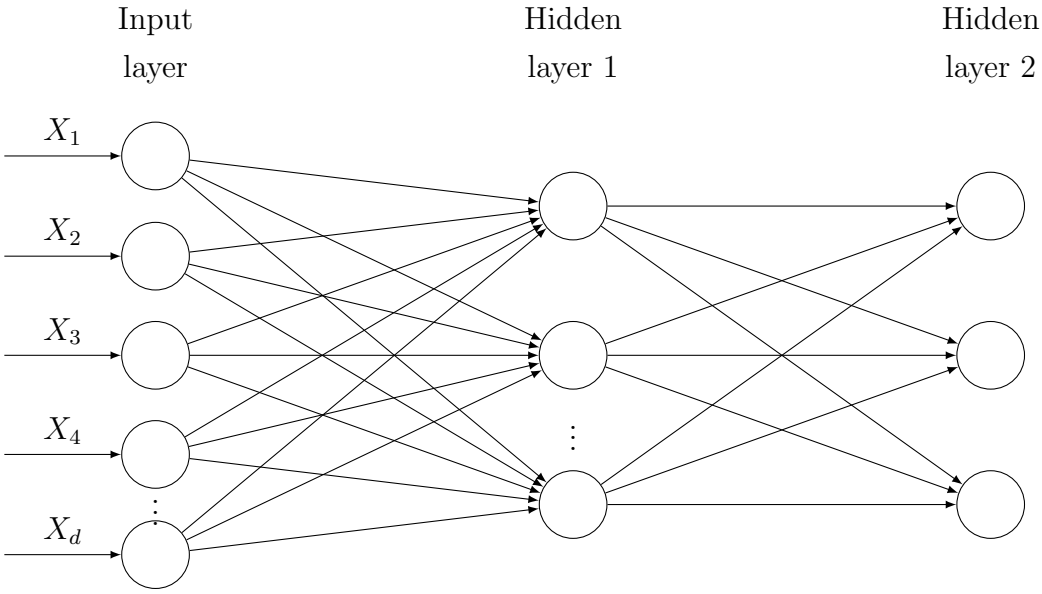
and these hidden layers feed their output to the final layer, which computes class scores and makes the final prediction:

$$f_k(X) = g_k(T), k = 1, ..., K$$

The parameters of ConvNets consist of weights α and bias terms β:

$$\{\alpha_{0m}, \alpha_m | m = 1, 2, ..., M\} M(p+1) \text{weights, and} \tag{7}$$

$$\{\beta_{0k}, \beta_k | k = 1, 2, ..., K\} K(M+1) \text{weights.} \tag{8}$$



A.3 The Back-Propagation Algorithm

This algorithm is how learning occurs. At each iteration of training, a combination of forward and backward passes make predictions, calculate error, and adjust weights to minimize error. During the forward pass, the algorithm works from input layer to output layer, calculating values for each neuron. After the output layer has produced predictions about what images belong to what class, the backward pass calculates total error $R(\theta)$ and how much each neuron from the last hidden layer contributed to each output neuron's error. Let $z_{mi} = h(\alpha_m^T x_i)$ and let $z_i = (z_{1i}, z_{2i}, ..., z_{Mi})$. Then the loss function $R(\theta)$ takes the form:

$$R(\theta) \equiv \sum_{i=1}^{N} R_i \tag{9}$$

$$= \sum_{i=1}^{N} \sum_{k=1}^{K} (y_{ik} - f_k(x_i))^2 \tag{10}$$

Where N is the number of inputs in the network and K is the number of classes. Then, the algorithm calculates how much each neuron from the previous layer contributed to the error of the last hidden layer, continuing layer-by-layer until it reaches the input layer. This pass propagates the error gradient backward [8] to compute the error gradient across all connected

[8]These calculations are done using reverse mode auto differentiation, the details of which are beyond the scope of this paper.

weights in the network. The error gradient vector is composed of partial derivatives that capture how much each weight (β_{km} and α_{ml}) contribute to total error:

$$\frac{\partial R_i}{\partial \beta_{km}} = -2(y_{ik} - f_k(x_i))g'_k(\beta_k^T z_i)z_{mi} \tag{11}$$

$$\frac{\partial R_i}{\partial \alpha_{ml}} = -\sum_{k=1}^{K} 2(y_{ik} - f_k(x_i))g'_k(\beta_k^T z_i)\beta_{km}\sigma'(\alpha_m^T x_i)x_{il} \tag{12}$$

$$\tag{13}$$

Finally, the backward pass uses the error gradient to adjust the weights toward a minimum – this is called the gradient descent step. At (r+1) iteration the gradient descent updates the weights via the following expressions:

$$\beta_{km}^{(r+1)} = \beta_{km}^r - \gamma_r \sum_{i=1}^{N} \frac{\partial R_i}{\partial \beta_{km}^{(r)}} \tag{14}$$

$$\alpha_{km}^{(r+1)} = \alpha_{km}^r - \gamma_r \sum_{i=1}^{N} \frac{\partial R_i}{\partial \alpha_{km}^{(r)}} \tag{15}$$

where γ_r is the learning rate – a tuning parameter that determines how "large" each gradient descent update should be – and α^r and β^r are the models' weights at the r^{th} iteration. We can rewrite equations 9 and 10 above to better understand how the algorithm propagates error backward through the network:

$$\frac{\partial R_i}{\partial \beta_{km}} = \delta_{ki} z_{mi} \tag{16}$$

$$\frac{\partial R_i}{\partial \alpha_{ml}} = s_{mi} x_{il} \tag{17}$$

$$\tag{18}$$

The value of δ_{ki} is error from units in the output layer and s_{mi} is the error from units in the hidden layers. By definition, these values satisfy the following condition:

$$s_{mi} = \sigma'(\alpha_m^T x_i) \sum_{k=1}^{K} \beta_{km} \delta_{ki}. \tag{19}$$

To summarize: the forward pass computes predicted values $\hat{f}_k(x_i)$ using fixed weights at each iteration. The resulting values are compared to target values Y_k in the backward pass, and error from each unit in the output layer $\delta_k i$ are calculated. The error gradient is then back propogated via equation 17 to each unit in the previous layers. Weights are adjusted via gradient descent and the process starts again.

Appendix B Automated Image Analysis

B.1 Detecting Race From an Image

Formulate the problem as it appears in Fu, He and Hou (2014)

B.2 Convolutional Neural Networks

Advances in computer science research have seen great success in image recognition using convolutional neural networks. These methods treat pixels as data and images as stacked matrices. Images are repeatedly collapsed until only the basic patterns of interest remain, represented by feature maps. Neural networks find statistical, weighted relationships between pixels and the machine learns to attribute the presence of an object to the patterns it sees over the course of several training epochs. Backward propagation and gradient descent algorithms are used to optimize weights, decreasing error in classification.

The work by Krizhevsky, Sutskever and Hinton (2012) represents the standard for image recognition tasks. The network relied on a ReLU nonlinearity function and eight layers (five convolutional layers and three fully-connected network layers). The CNN achieved a top-1 error and top-5 error rates of 37.5% and 17.0% on a dataset of 1.2 million images. The images were part of the ImageNet LSVRC-2010 contest in which high-resolution images must be classified into 1000 different classes. The authors used a parallel architecture to train on multiple GPU's, increasing computing performance. The shere amount of images and classes far outweigh any task that political science has, in terms of classification problems. The success of this CNN provides researchers a template with which to build their own classifiers; more than that, the rise of transfer learning (detailed below) decreases computing costs adapting existing architectures to new learning tasks.

Ciregan, Meier and Schmidhuber (2012) advocate the use of unsupervised, deep convolutional neural networks using a multi-column framework. The authors also rely on GPU's to increase performance. The multi-column architecture reduce the need for preprocessing and decrease over-fitting. The approach relies on several different neural networks running in parallel, the predictions of which are averaged across columns. The authors achieve near human performance on the MNIST digits dataset. This is another instance of deep CNN's performing well in image classification. Work by Simonyan and Zisserman (2014) also demonstrates the importance of depth for performance in image classification.

Guo et al. (2016) offer an in-depth review of deep learning methods more broadly. The authors highlight several different architectures that have the highest performance in the ILSVRC classification competition. This includes the AlexNet architecture (Krizhevsky,

Sutskever and Hinton, 2012) that became a template for many CNN architectures that followed; the Clarifai (Zeiler and Fergus, 2014) model highlighted the importance of intermediate feature layers; SPP (He et al., 2014) proposed "spatial pyramid pooling" to better deal with image resolution; VGG (Simonyan and Zisserman, 2014) examines the importance of depth to classification performance; and GoogLeNet (Szegedy et al., 2015) pushes depth and width limits without increasing computational requirements.

While this may seem overly technical and unimportant to political scientists, this (brief) review indicates the various advancements that have been made in recent years to improve classification performance and decrease computational requirements. The architecture of CNN's vary greatly – no two networks are built alike – and as image analysis grows in prominence in the discipline, it is vital that scholars understand the technical details of their computational methods. As the next section goes to show, understanding CNN architecture is key to adapting the method for research purposes.

Convolutions

The convolution layer applies a square kernel that rotates across the image, detecting patterns in the pixels. The earlier layers detect low-level, simple patterns (splotch of colors, line edges) while the deeper layers detect higher-level objects (like cars, fire, or groups of people). This layer takes the original 300 x 300 image as input and produces a feature map of "neurons" as output. These neurons are connected to local regions (as opposed to the fully connected, dense networks seen in the deeper layers). Neurons are the dot products between weights and local pixels. The size of the filter (and the resulting feature map) is a user-defined hyper-parameter.

B.2.1 Pooling

The pooling layer reduces the dimensionality of the image data along the width and height dimensions [9]. *MaxPooling* refers to a procedure where groups of pixels are reduced to singular values, i.e., the pixel with the highest value. This makes sense logically, when you think of the aspects of an image that offer the most information about its contents. In trying to train a machine to recognize fire, you'd want to highlight pixels that are the brightest yellow. Usually, a pooling layer occurs after each convolution layer.

[9]Note that the convolution process expands the depth of the input object, while leaving width and height unchanged. The pooling process reduces the dimensionality of width and height, while leaving depth unchanged.

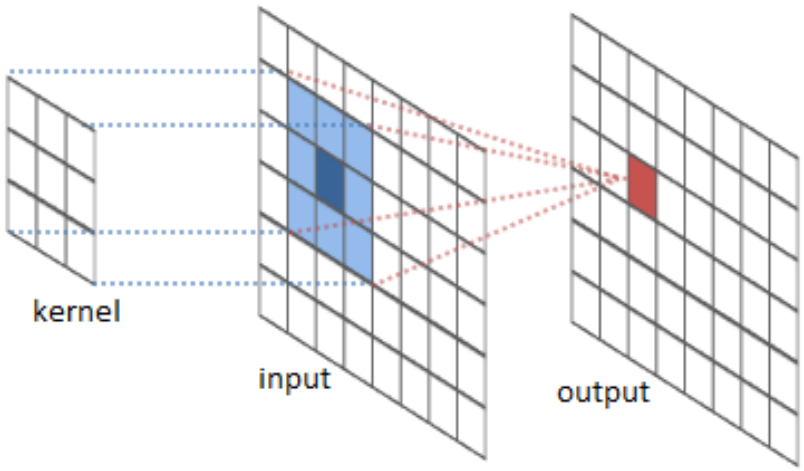


Figure 13: The kernel takes groups of pixels as input and computing their weighted sum as output (taken from: http://colah.github.io/posts/2014-07-Understanding-Convolutions).

Term	Notation	Definition
Error Function	$R(\theta)$	Difference between target values and predicted values
Classes	$k = 1, ..., K$	Number of classes
Nodes	$m = 1, ..., M$	Number of nodes for a given layer
Vector of outputs	T	778
Output function	$g_k(T)$	Usually softmax
Activation function	$\sigma(v)$	Computation to determine output of given node, usally ReLU
Weights	α_m and β_k	Parameters of the model, learned via back propagation
Hidden Layers	Z_m	Linear combination of inputs and weights
Predicted Values	$\hat{f}_k(x_i)$	
Error (output layer)	δ_k	Errors from output layer for class k
Error (hidden layer)	s_m	Error from hidden layer for unit m

Table 3: Table to test captions and labels

B.3 Problem One: Training Data

The immediate problem with CNN's is the amount of resources they take to train from scratch. Not only do networks require computing power (many frameworks leverage graphics processing units, or GPU's, to improve efficiency and decrease training time) but they also require thousands of training images per class – in addition to the test images needed for cross validation. Many of the top performing networks take weeks to train. If applied to a political science problem, gathering enough training data and dividing images into classes would hardly be beneficial – why not just conduct the study with humans coding the images? This problem seemingly presents a huge barrier for computer vision to become a mainstream method in social science, but the concept of *transfer learning* offers a solution.

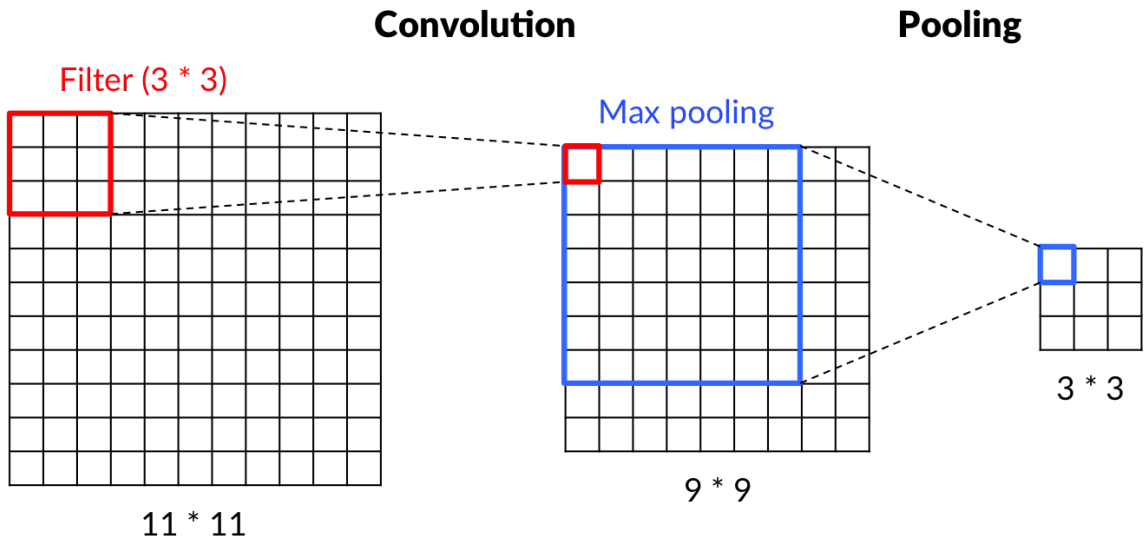


Figure 14: (taken from: https://saitoxu.io/2017/01/01/convolution-and-pooling.html).

This procedure transfers mid-level knowledge from a *source task* to a *target task*. In the early layers of a network, the machine is learning the basic features common to any photograph: edges, borders, and shapes. After each iteration – after the network has seen the training set again – the machine detects the more complicated patterns that are specific to each class. If images are simply stacks of matrices containing pixel values, then learning to differentiate between a picture of a White man and a Black woman is a matter of re-arranging values in a 4-dimensional tensor.

For instance, Oquab et al. (2014) use the network architecture from Krizhevsky, Sutskever and Hinton (2012) but remap the learned parameters onto a new set of training images. Specifically, Oquab et al. (2014) use ImageNet-trained layers of CNN to classify images in Pascal VOC. The convolutional network is first trained on 1.2 million images and 1000 classes from the ImageNet 2012 Large Scale Visual Recognition Challenege (ILSVRC-2012). The authors then transfer mid-level features learned by the network to the Pascal VOC 2007 and 2012 object classification tasks using only limited data. The success of this technique demonstrates the viability of transfer learning for CNN's to train on limited data.

References

Abraham, Linus and Osei Appiah. 2006. "Framing news stories: The role of visual imagery in priming racial stereotypes." *The Howard Journal of Communications* 17(3):183–203.

Brader, Ted, Nicholas A Valentino and Elizabeth Suhay. 2008. "What triggers public opposition to immigration? Anxiety, group cues, and immigration threat." *American Journal of Political Science* 52(4):959–978.

Brantner, Cornelia, Katharina Lobinger and Irmgard Wetzstein. 2011. "Effects of visual framing on emotional responses and evaluations of news stories about the Gaza conflict 2009." *Journalism & Mass Communication Quarterly* 88(3):523–540.

Ciregan, Dan, Ueli Meier and Jürgen Schmidhuber. 2012. Multi-column deep neural networks for image classification. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*. IEEE pp. 3642–3649.

Corrigall-Brown, Catherine and Rima Wilkes. 2012. "Picturing protest: The visual framing of collective action by First Nations in Canada." *American Behavioral Scientist* 56(2):223–243.

Dahmen, Nicole S. 2012. "Photographic Framing in the Stem Cell Debate: Integrating Eye-Tracking Data for a New Dimension of Media Effects Research." *American Behavioral Scientist* 56(2):189–203.

Dixon, Travis Lemar and Daniel Linz. 2000. "Overrepresentation and underrepresentation of African Americans and Latinos as lawbreakers on television news." *Journal of communication* 50(2):131–154.

Fox, Cybelle. 2004. "The changing color of welfare? How Whites' attitudes toward Latinos influence support for welfare." *American Journal of Sociology* 110(3):580–625.

Fu, Siyao, Haibo He and Zeng-guang Hou. 2014. "Learning Race from Face : A Survey." *IEEE Transactions on Pattern Analysis and Machine Intelligence* 36(12):2483–2509.

Gilens, Martin. 1996. "Race and Poverty in America Public Misperceptions and the American News Media." *Public Opinion Quarterly* 60(4):515–541.

Gilens, Martin. 2009. *Why Americans hate welfare: Race, media, and the politics of antipoverty policy*. University of Chicago Press.

Grimmer, Justin and Brandon M Stewart. 2013. "Text as data: The promise and pitfalls of automatic content analysis methods for political texts." *Political analysis* 21(3):267–297.

Guo, Yanming, Yu Liu, Ard Oerlemans, Songyang Lao, Song Wu and Michael S Lew. 2016. "Deep learning for visual understanding: A review." *Neurocomputing* 187:27–48.

Hancock, Ange-Marie. 2004. *The politics of disgust: The public identity of the welfare queen*. NYU Press.

He, Kaiming, Xiangyu Zhang, Shaoqing Ren and Jian Sun. 2014. Spatial pyramid pooling in deep convolutional networks for visual recognition. In *European Conference on Computer Vision*. Springer pp. 346–361.

Iyer, Aarti and Julian Oldmeadow. 2006. "Picture this: Emotional and political responses to photographs of the Kenneth Bigley kidnapping." *European Journal of Social Psychology* 36(5):635–647.

Kinder, Donald R and Shanto Iyengar. 1987. "News that matters: Television and American opinion." *Univeristy of Chicago Press, Chicago: IL* .

Krizhevsky, Alex, Ilya Sutskever and Geoffrey E Hinton. 2012. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*. pp. 1097–1105.

McCulloch, Warren S and Walter Pitts. 1943. "A logical calculus of the ideas immanent in nervous activity." *The bulletin of mathematical biophysics* 5(4):115–133.

Mendelberg, Tali. 2001. *The race card: Campaign strategy, implicit messages, and the norm of equality*. Princeton University Press.

Oquab, Maxime, Leon Bottou, Ivan Laptev and Josef Sivic. 2014. Learning and transferring mid-level image representations using convolutional neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. pp. 1717–1724.

Pan, Sinno Jialin and Qiang Yang. 2010. "A survey on transfer learning." *IEEE Transactions on knowledge and data engineering* 22(10):1345–1359.

Russakovsky, Olga et al. 2015. "ImageNet Large Scale Visual Recognition Challenge." *International Journal of Computer Vision (IJCV)* 115(3):211–252.

Russell Neuman, W, Lauren Guggenheim, S Mo Jang and Soo Young Bae. 2014. "The dynamics of public attention: Agenda-setting theory meets big data." *Journal of Communication* 64(2):193–214.

Simonyan, Karen and Andrew Zisserman. 2014. "Very deep convolutional networks for large-scale image recognition." *arXiv preprint arXiv:1409.1556* .

Singh, Vivek Kumar, Saket Hegde and Akanksha Atrey. 2017. Towards Measuring Fine-Grained Diversity Using Social Media Photographs. In *ICWSM*. pp. 668–671.

Soroka, Stuart, Mark Daku, Dan Hiaeshutter-Rice, Lauren Guggenheim and Josh Pasek. 2017. "Negativity and Positivity Biases in Economic News Coverage: Traditional Versus Social Media." *Communication Research* p. 0093650217725870.

Szegedy, Christian, Vincent Vanhoucke, Jonathon Shlens and Zbigniew Wojna. 2014. "Rethinking the Inception Architecture for Computer Vision.".

Szegedy, Christian, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke and Andrew Rabinovich. 2015. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. pp. 1–9.

Valentino, Nicholas A, Ted Brader and Ashley E Jardina. 2013. "Immigration opposition among US Whites: General ethnocentrism or media priming of attitudes about Latinos?" *Political Psychology* 34(2):149–166.

Valentino, Nicholas A, Vincent L Hutchings and Ismail K White. 2002. "Cues that matter: How political ads prime racial attitudes during campaigns." *American Political Science Review* 96(1):75–90.

Wang, Xiaolong, Rui Guo and Chandra Kambhamettu. 2015. Deeply-learned feature for age estimation. In *Applications of Computer Vision (WACV), 2015 IEEE Winter Conference on*. IEEE pp. 534–541.

Wang, Yu, Haofu Liao, Yang Feng, Xiangyang Xu and Jiebo Luo. 2016. "Do they all look the same? deciphering chinese, japanese and koreans by fine-grained deep learning." *arXiv preprint arXiv:1610.01854* .

Wang, Yu, Yang Feng and Jiebo Luo. 2017. Gender politics in the 2016 US presidential election: a computer vision approach. In *International Conference on Social Computing,*

Behavioral-Cultural Modeling and Prediction and Behavior Representation in Modeling and Simulation. Springer pp. 35–45.

Wang, Yu, Yuncheng Li and Jiebo Luo. 2016. Deciphering the 2016 US Presidential Campaign in the Twitter Sphere: A Comparison of the Trumpists and Clintonists. In *ICWSM*. pp. 723–726.

Won, Donghyeon, Zachary C Steinert-Threlkeld and Jungseock Joo. 2017. "Protest Activity Detection and Perceived Violence Estimation from Social Media Images." *arXiv preprint arXiv:1709.06204* .

Young, Lori and Stuart Soroka. 2012. "Affective news: The automated coding of sentiment in political texts." *Political Communication* 29(2):205–231.

Zeiler, Matthew D and Rob Fergus. 2014. Visualizing and understanding convolutional networks. In *European conference on computer vision*. Springer pp. 818–833.

Zintgraf, Luisa M, Taco S Cohen, Tameem Adel and Max Welling. 2017. Visualizing Deep Neural Network Decisions: Prediction Difference Analysis. In *Internation Conference on Learning Representations*.