A Fast, Cost-Efficient Image Annotation Algorithm with an Application to Analyzing the Visual Components of Protests*

Marco Morucci

Abstract

Image data is an important source of information for political scientists, as it can be used to measure and extract important theoretical concepts. However, existing methods for automated image analysis require massive amounts of hand-labelled examples to train on. This paper proposes an automated methodology to extract information, like presence or absence of certain elements in an image, from large datasets with the need for only a small amount of human-annotated images containing the desired elements. Extracting information by hand labelling large datasets is slow and expensive, and the proposed methodology will dramatically reduce both time and monetary costs of this process. The proposed approach is based on a state-of the-art deep learning framework, and is applied to the task of detecting images of protests among a larger set of social media images, showing that images of protests can be detected with around 15% error from only 200 labels.

1 Introduction

The importance of non-numeric data sources in quantitative political science has grown exponentially in the last 10 years. Political scientists have been employing image, audio, video and text

*Preliminary draft: please don't circulate or cite without permission.

data to draw empirical conclusions more and more frequently. This is for two good reasons: first images, text and video have direct effects on public attitudes that must be taken into account to fully understand modern political developments. Second, non-traditional data sources can lead to better measurement of many long-standing quantities of interest in political science, such as political ideology (Xi et al., 2020) or wealth (Weidmann and Schutte, 2017), in turn making for more reliable testing of new and existing theories.

Text-as-data is a great example of how methodologies for non-numeric data have had a positive impact on the field. Earlier developments of tools for cheap and reliable analysis of text data (e.g., Roberts et al., 2016) enabled many social scientists to measure new and old quantities of interest, and to develop and test theories more reliably (Grimmer and Stewart, 2013). Examples of how text as data has been used to get at previously hard to measure concept include variation in ideological similarity between voters and politicians (Barberá et al., 2019), and meaning of political concepts (Rodman, 2020). Following the successful example of text as data, this paper proposes a methodology for fast, and cheap annotation of image data, aimed at enabling social science researchers to use images for measurement and theory-testing in a way similar to text data.

Unfortunately, summarizing the content of images in ways amenable to quantitative analysis is a substantially harder problem than doing the same with text (Joo and Steinert-Threlkeld, 2018). Because of this, analysis of image data has remained less accessible to a vast groups of social science researchers. Traditional analyses of image content have often relied on human annotation of large corpora of images. For example, both Won et al. (2017), and Casas and Williams (2019) rely on Amazon M-turk workers to annotate databases of more than 10 thousand images to study the visual components of protest. While human annotation is reliable and precise, it is also expensive and time-consuming: researchers willing to work with image data have to have both the monetary and time resources to have workers look at each image in their database, and code it accordingly.

In this paper we propose an algorithm for annotating large corpora of images, with the need of only a small number of human-labelled examples. For example, in our application we can annotate around 40 thousand social media images as either depicting public rallies or not from

only 200 human-labelled examples, and with an error rate of around 15%. We leverage recent advances in Semi-Supervised Learning with image data in the construction of our algorithm, and show that it performs either on par with, or better than existing technology for the same problem. Most importantly, our methodology is also entirely implemented using open source software and freely available cloud computing resources. This bypasses the need for expensive computational resources often needed to train large deep learning models for images in reasonable amounts of time, making image data accessible also to researchers who do not have easy access to the resources needed to train a large deep learning model.

Together, the low requirement for expensive and time consuming hand-annotated examples and costly computational resources make our approach easily accessible to most researchers interested in analyzing political images, and enables researchers with the resources to hand-label large amounts of images to run a simple and effective pilot study on their data.

We apply our methodology to the problem of extracting the presence/absence of certain visual components such as police, fire, large crowds from images of protests shared on social media. This is an important problem, as there is evidence that images affect both public perceptions of (Torres, 2020), and participation (Casas and Williams, 2019) in public rallies. The dataset we use has also been manually annotated by human coders (Won et al., 2017), and as such we can validate the performance of our methodology on it. We show that our methodology attains good levels of accuracy with the need of only a small amount of human-coded examples.

Our paper will proceed as follows: first, we review literature on the importance of images in political contexts, as well as summarize the various existing barriers to entry in the analysis of image data. Second, we briefly introduce the techniques and principles of deep learning that will guide the development of our algorithm. Third, we introduce our algorithm, and give full training configurations. Finally, we apply our proposed method to the extraction of visual components from images of protests.

1.1 The Importance of Images in Political Contexts

Political science has long since observed the impact of images and visual media on various aspects of politics. Here we make the case that images are a powerful source of data both for measurement of quantities of interest, and for understanding the impact that visual media has on public attitudes.

Starting from the latter, a substantial amount of work has been done that shows how visual media can sway public attitudes: For example, a large body of literature has shown that politicians strategically choose their appearance to potentially affect voters (Lawson and McCann, 2005; Lawson et al., 2010; Joo et al., 2014; Hayes et al., 2014; Mattes and Milazzo, 2014; Anastasopoulos et al., 2016; Milazzo and Mattes, 2016), and additional research has confirmed that these visual choices do indeed matter: electoral outcomes are related to various visual cues from candidates (Sullivan and Masters, 1988; Todorov et al., 2005; Horiuchi et al., 2012; Joo et al., 2015). Aside from candidate cues, the media has been shown to possess the capacity to sway public attitudes on a variety of topics through the visuals they present to viewers (Powell et al., 2015). Areas in which images have been observed to affect public preferences range from elections (Gilliam Jr and Iyengar, 2000; Druckman and Parkin, 2005; Grabe and Bucy, 2009) and climate change (O'Neill and Smith, 2014), to police interactions (Makin et al., 2019), public movements (Won et al., 2017; Casas and Williams, 2019; Torres and Cantú, 2020) and international affairs (Hansen, 2015). Additionally, there's evidence from both lab and field experiments that individuals develop attitudes about others based on visual cues, and that these attitudes affect their political decision-making (Tingley, 2014; Hainmueller and Hangartner, 2013). These are only some of the many examples of how images have been shown to matter for political attitudes: if we wish to improve our understanding of public opinion we have to take visual data into account.

In addition to being able to capture the direct effect that visual elements have on public attitudes, image data is a powerful way to measure other quantities potentially of interest to political scientists. Existing work has successfully employed image data to measure concepts such as race (Sen and Wasow, 2016), political ideology (Xi et al., 2020), emotional affect (Rheault and Borwein, 2019), violence (Won et al., 2017), wealth (Weidmann and Schutte, 2017), various de-

mographic characteristics (Gebru et al., 2017), and occurrence of protest events (Zhang and Pan, 2019). These examples should show that image data can be useful for measurement of many existing and new political science variables, often at a much more fine-grained level than what other types of data allow.

In sum, we have ample evidence that image data is meaningful, prominent, and useful in political science. Unfortunately, there is a substantial startup cost to the analysis of such data, a cost that this paper aims to lower.

1.2 The time, monetary, and computational cost of analyzing images

While existing work attests to the promise of visual data for political science research, we contend that there are some important barriers to entry to visual data analysis for social science researchers.

First, detecting the presence/absence of the element of interest in images requires human annotation of at least a sufficient amount of data to train a deep learning model to label additional inputs with good accuracy. Unfortunately, the amount of data needed to train a fully supervised model to accurately predict visual elements out of sample is well in the tens, if not hundreds, of thousands, due to the high dimensionality of image data (Zhang et al., 2016). While it is true that, once trained, a model trained to extract one concept from images can be applied to many settings, it is also true that there are many and varied concepts potentially measurable from image data for bespoke supervised models to eventually be available for all of them. In addition, researchers may always propose new concepts that could be measurable with images: it would be unreasonable to ask researchers without the resources for image data analysis to wait for others with the necessary resources to train a model for their specific variable.

Unsupervised methods are unfortunately not (yet) viable to extract information from images to the level of accuracy needed for social science. This is the main difference between image-as-data and text-as-data in political science: advancements in unsupervised methods for the latter have made text analysis of good enough quality easily accessible to most researchers (Grimmer and Stewart, 2013). This is not yet possible for image data. State-of-the art unsupervised encod-

ing method such as Variational Auto Encoders (VAE) (Kingma and Welling, 2013) can produce near lossless low-dimensional representations of the input images, but these representations are uninterpretable: researchers are not yet able to associate any of these dimensions with concepts of theoretical interest (Higgins et al., 2017). On the other hand, methodologies based on manual feature extraction that do not rely on deep learning are incapable of achieving the level accuracy needed for meaningful measurement of concepts of interest to political scientists.

This leaves researchers with the need to hand-label a large amount of images, either with the goal to train a fully-supervised deep network, or for direct analysis. Manually labelling tens of thousands of images would take a single researcher an exorbitant amount of time, while outsourcing hand-labelling to contracted workers requires a monetary investment that not all researchers interested in political images can make.

Finally, there is one additional cost that researchers face when wanting to implement deep-learning techniques: the computational resources needed to train a modern very deep model such as ResNet or Inception from scratch are massive: a high-performance cluster of GPUs is often needed to fully train a model in a reasonable time-frame. While it is true that pre-trained model can be adapted to a specific task via fine-tuning at a much lower computational cost, these models are always fully supervised, and require a large amount of labelled data.

This paper aims at introducing a methodology that lowers all three types of cost introduced before: we will propose a model that can be trained end-to-end with a small amount of human-annotated examples, and in a reasonable time-frame on freely available cloud computing resources. This in turn, will work towards making image data analysis more accessible to a large group of researchers.

2 Semi-Supervised Image Classification in Machine Learning

Here we introduce and summarize the building blocks of our methodology: Convolutional Neural Networs (CNNs), which have become the standard for analyzing image data, and Semi-Supervised

Learning (SSL), which has been successfully applied to problems in image classification (Krizhevsky et al., 2012). We give a brief introduction to both CNNs and SSL, for comprehensive introductions to CNNs aimed at social science, see Joo and Steinert-Threlkeld (2018), Williams et al. (2020), and Torres and Cantú (2020); for an introduction to SSL see Zhu and Goldberg (2009).

Before moving on to these building blocks, we introduce some notation and clarify some terminology that will be used throughout the paper. We will be working with a large set of unlabeled images $\mathbf{X}^u = \{\mathbf{x}_i^u\}_{i=1}^{n^u}$, and a small set of labeled images $\mathbf{X}^l = \{\mathbf{x}_i^l\}_{i=1}^{n^l}$, the latter associated with one-hot encoded labels, i.e., categorical labels for n^c categories that have been expanded into a binary vector of dummy variables: $\mathbf{Y}^l = \{\mathbf{y}_i^l\}_{i=1}^{n^l}$, where each $\mathbf{y}_i^l$ is a vector of zeros and ones of length n^c. We use $\mathbf{x}$ to refer to an arbitrary image in either set, and we assume that all images are real-valued arrays with dimensionality of $W \times H \times 3$ pixels, where the last dimension represents the standard red, green and blue color channels that digital images usually have. As it is common in the machine learning literature, we will refer to input images as input features (or just features), where in this case a single feature is a pixel in one of the input images.

2.1 Convolutional Neural Networks

Virtually all of modern machine learning methods for image data employ some variant of the CNN algorithm, this is, in order, an extension of the Deep Neural Networks. A Deep Neural Network (DNN) is a prediction algorithm based around the interaction of many different learned features for the input data. In brief, a DNN works by feeding input images to many intermediate nodes (known as neurons), each of which computes a set of weights for each of the input features, and then applies a non-linear transform to the product of features and weights. The outputs of each neuron are then concatenated together and fed as input to every other neuron in a subsequent layer of the DNN. The presence of many consecutive layers is what makes networks deep. At an intuitive level DNNs are capable of learning complex functions of the inputs because they simulate interactions between input features with interactions between neurons.

What is a CNN Introduced in LeCun et al. (1989, 1995), CNNs are an extension of the NN algorithm tailored specifically to 2-dimensional inputs, such as images, in which intermediate layers do not use a simple linear combination of inputs, but instead rely on the convolution operator, which for discrete data such as images, is defined as follows:

$$F(w,h) = \sum_{c=1}^{C} \sum_{i=1}^{W^K} \sum_{j=1}^{H^K} x(w-i-1, h-j-1, c) K(i,j). \tag{1}$$

Here $x(i,j,c)$ refers to the pixel at location i,j and on channel c of an arbitrary image, and K is a $W^K \times H^K$ matrix of real-valued weights, commonly known as *kernel*. This operation essentially amounts to creating many local linear combinations of regions of size $W^K \times H^K$ of each image, where the weights are defined by the kernel. These local combinations are then stacked together in a matrix to create an intermediate representation of the input image known as a feature map. The feature map should be a less noisy representation of the input image. We usually would like feature maps to be of smaller size than inputs, so we usually evaluate $F(w,h)$ only at certain pixel locations in the input, the amount of pixels which we skip every time before computing a convolution is known as *stride*. In tandem with stride, it is also common practice to use multiple kernels to compute a convolution at each location: if we have n^K kernels, denoted by $(K_1, \ldots, K_k, \ldots K_{n^K})$, we will compute the quantity in (1) with each one, and stack the results on the third dimension of the resulting array. The resulting feature map, $F(w,h,k)$, will then have a number of channels equal to n^K. Finally, the feature map is taken as input for the following layer of the CNN: the convolution operation is applied to it as described before to generate yet another feature map, which should, ideally, be smaller in terms of dimensionality, but contain more meaningful information. This process is repeated until the last layer of the CNN, at which point a classification head is applied to the last learned feature map to map the learned features ot output labels. This classification head is usually a linear combination of the features produced by the convolutional layers. There are two additional important operations that are usually applied to each feature map before it is used as input to an additional layer: first *nonlinear activation*. Each value in the feature map is fed

to a non-linear activation function, popular choices of which are sigmoid, ReLU, or tanh. Second, *pooling* is an operation that aggregates activation values in a region of the feature map, either by averaging, summing, or taking the max. For example, a 2x2 max pooling layer would scan all 2x2 regions of the feature map and take the max of all such regions, thus constructing a substantially smaller feature map. This operation helps eliminate spatial correlation between pixels, and enforce location invariance of the learned features. Figure 1 summarizes the typical CNN architecture just discussed.

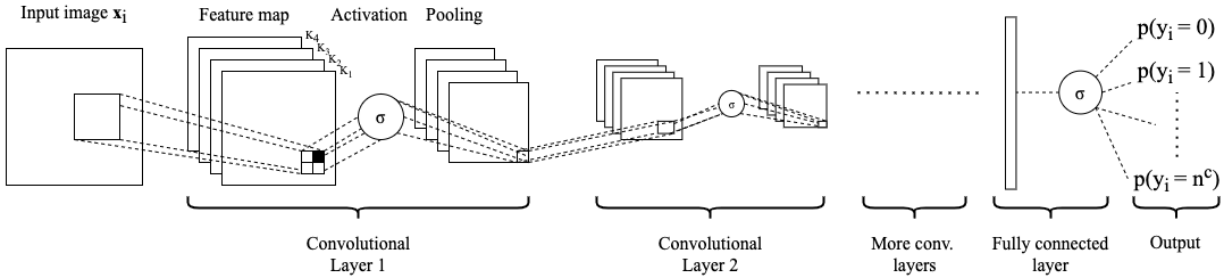

Figure 1: Typical structure of a CNN. The inner square represents an arbitrary region of the input image. The black square in the feature map block shows how max pooling preserves only one of the four regions considered. Convolutional features are flattened into a 1-d vector before being passed into the final fully connected layer.

It has been shown that CNNs excel in image-based learning tasks such as image classification (Krizhevsky et al., 2012) and object detection (Girshick et al., 2014; Lin et al., 2017), which are areas adjacent to our application of interest here.

Training CNNs Deep learning models are trained to minimize a loss function usually consisting of some form of prediction error. For example, a classic loss for supervised classification is cross-entropy, which is defined as follows:

$$\ell_{CE} = -\sum_{i=1}^{n^l}\sum_{c=1}^{n^c} y_c^l \log(g(\mathbf{x}_i^l)),$$

where $g(\mathbf{x}_i^l)$ is the prediction output by the CNN for labeled input image i. A model that minimizes this loss will predict well on new data points. Solving this minimization problem for a

model as complex as a CNN is not at all easy. Fortunately, however, all operations that make up CNN layers are *differentiable*. Therefore, gradient-based optimizers such as gradient descent can be used to iteratively minimize the loss function of interest, with guarantees on optimality of the solution. Datasets used to train CNNs are often too large to fit in memory, and in general, computing gradients on the entire dataset is intractable. Because of this, instead of using the whole data at each training step of the gradient descent procedure, a much smaller random sample of the dataset known as a *minibatch* is employed. Gradients computed on minibatches of data are unbiased and consistent estimates of gradients computed on the whole data, for most common loss functions, and therefore lead to approximately optimal gradient descent updates. A gradient descent optimizer that uses minibatches of data is known as Stochastic Gradient Descent (SGD).

Other important variables and techniques that come into play when training a large CNN are the *learning rate*, the scaling factor applied to each gradient at each update, which is usually decayed as training progresses. *Weight decay*, a scaling factor applied to weights at each update, also decayed as train progresses, and intended to regularize weight updates. *Exponential moving averages* of the weights at each iterations, which are used to mitigate the randomness inherent in computing gradients on smaller subsets of the data: a gradient update with this technique consists of a linear combination of the weights at the last iteration and the newly updated weights. As a simple and abstract example, the value of the weights of a CNN at training step t , denoted by $\boldsymbol{\theta}$ are given by:

$$\boldsymbol{\theta}_t = \boldsymbol{\theta}_{t-1} + \eta_t v_t,\ v_t = \beta v_{t-1} - \nabla \ell(\boldsymbol{\theta}_{t-1}) - w_{t-1}\boldsymbol{\theta}_{t-1}, \tag{2}$$

where η_t is the learning rate at iteration t, β is the exponential moving average parameter, $\nabla \ell(\boldsymbol{\theta}_{t-1})$ is the gradient of the loss with respect to the weights being updated, and w_{t-1} is the weight decay parameter. Note, finally, that many more optimization techniques exist that apply gradients (ADAM, RMSProp) with different formulas than the one in Eq. (2), but the main framework remains that of stochastic gradient descent.

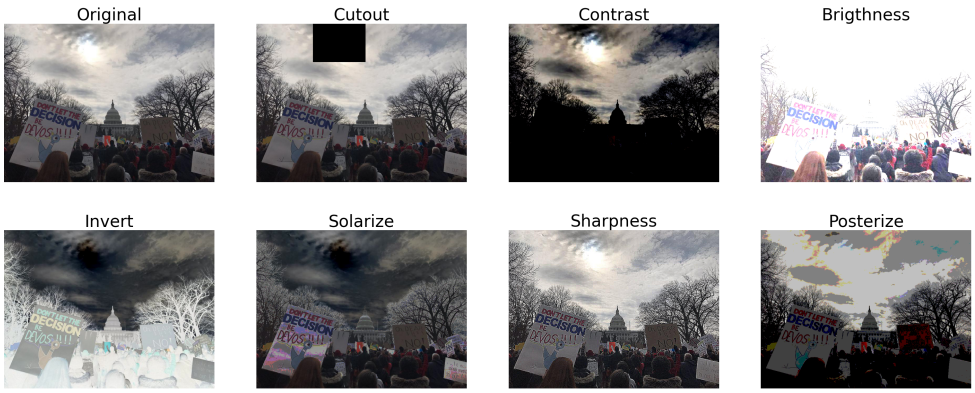

Figure 2: Some sample augmentations applied to an example image from the protests dataset.

2.1.1 Data Augmentation

Data augmentation is one of the key strategies of modern deep learning for image data (Wang et al., 2017). In brief, input images are transformed in one or more ways and then the model is trained on the transformed version of the input. Common transformations include rotations, scaling, cropping regions, altering colors and adding noise or blur to input images. See Shorten and Khoshgoftaar (2019) for a comprehensive list of augmentations currently in use in the deep learning literature. Figure 2 presents some sample augmentations that we will use in our framework.

Augmentations to be applied to a specific input image are often randomly chosen from a larger set, and parameters for these augmentation functions (i.e., the degrees of a rotation, or the radius of a gaussian blur) are also chosen randomly (Xie et al., 2019). This implies that two augmented versions of the same input image will rarely be the same. This can be used to improve the learned feature maps by forcing the CNN to learn similar feature maps for different augmented versions of the same input image. Other modern augmentation techniques learn the augmentations and parameters that are most effective during training (Cubuk et al., 2019; Berthelot et al., 2019a) by optimizing related quantities or cross-validation.

2.2 Semi-Supervised Deep Learning

Semi-Supervised Learning is the problem of leveraging a usually small amount of labelled data together with a larger amount of unlabeled data to train a machine learning model to predict the labels. There are two main applications of SSL: *inductive* SSL, where a model is trained on both unlabeled and labeled data, and then used to predict labels on a new, unseen dataset, and *transductive* SSL, in which a model trained on both sets is used to predict labels on the unlabeled data.

2.2.1 Self-Training

There are two main guiding principles of SSL that we will employ in building our framework. The first is *self-training* or pseudo-labelling. This is the practice of labelling subsets of the unlabeled data during train time, and then treating those subsets as labelled in subsequent training iterations. A self-training loop would consist of roughly the following three steps:

1. Train the model on the labeled data

2. Predict labels for the unlabeled data

3. Remove a high-confidence subset from the unlabeled data and add it to the labeled data.

The loop above would be repeated until no more unlabeled data is left. In our algorithm, we will implement this idea in a soft way (Lee, 2013): instead of choosing an arbitrary threshold of prediction confidence, beyond which we consider a training point as labelled, we will use predicted class probabilities directly as weights for the unsupervised portion of the overall training loss.

Self-training has been shown to be equivalent to minimizing entropy between the learned class label distributions (Amini and Gallinari, 2002; Grandvalet and Bengio, 2005). This takes advantage of the separation between the labels in covariate space to leverage the unlabeled data for learning a good classifier.

2.2.2 Consistency

The second important principle of SSL that we will follow in our procedure is the idea of consistency of the learned representations: if the model makes similar predictions for two inputs, then the inputs should be similar to each other. In the case of CNNs, this means that the representations – the feature maps – learned by the model for images with similar labels should also be similar. This concept has been a mainstay of SSL, with several trademark SSL algorithms, such as semi-supervised SVMs, cluster-then-label methods, and graph-based methods relying heavily on it (Zhu and Goldberg, 2009). The main idea behind this principle, depicted in Figure 3 is that, if inputs with similar classes are close in feature space, then we can use this proximity to extend labels from labeled examples to unlabeled ones.

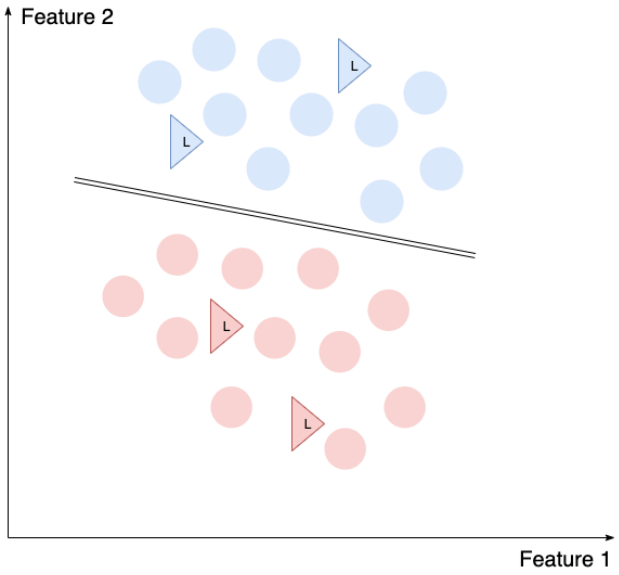

Figure 3: Consistency in SSL. In this simple example we have two classes, red and blue, and two features. Labeled instances are represented as triangles and unlabeled instances as circles. Points from the red class and the blue class are clustered together in roughly the same region of the input feature space, making it possible to use consistency to extend labels from labeled instances to unlabeled instances that are close in input space.

In deep learning, consistency is often implemented via *data augmentations*: the same input image is distorted or modified in different ways and then fed to the model. The model is then encouraged to create similar representations for different augmented versions of the same input

image (Sajjadi et al., 2016; Berthelot et al., 2019b,a). Forcing the model to align the distributions of similar examples is an instance of entropy regularization (Grandvalet and Bengio, 2005): if examples with similar labels are indeed clustered in feature space, then minimizing the entropy of model representations in similar parts of the space will also result in similarly clustered predictions. Enforcing consistency has been shown to improve CNN performance in many different SSL tasks (Tarvainen and Valpola, 2017; Berthelot et al., 2019a; Chen et al., 2020b), and we take advantage of this idea in our algorithm.

3 Proposed Algorithm

Our approach for semi-supervised classification relies on incorporating all the insights from semi-supervised deep learning discussed above into a classification framework. We wish to propose a model that relies on consistency regularization, and self-training in order to leverage both labeled and unlabeled data to improve prediction accuracy. We do so by combining simple and effective data augmentations (Sohn et al., 2020) with techniques from self-training via consistency regularization known as contrastive learning (Chen et al., 2020a,b).

We propose an algorithm with the following components:

1. Two separate randomized data augmentation procedures, a weak augmentation procedure and a strong augmentation procedure

2. A CNN model that constructs representations of input images

3. A classification head taking as inputs constructed representations and outputting class predictions

4. A loss function that trades off supervised prediction accuracy with consistency regularization and soft self-training.

We describe our approach at each of these steps.

Strong and weak data augmentation Following Sohn et al. (2020), we use two data augmentation procedures. First, we implement a *weak augmentation* that consists of randomly flipping images on the horizontal axis, with a probability of .5 and a random translation of at most 12.5% on either axis. Second, we implement a *strong augmentation* that consists of applying the RandAugment algorithm (Cubuk et al., 2020) to the input data. This algorithm chooses between 2 and 6 transformations from a list of 14 possible ones uniformly at random (the number of transformations chosen is also randomized), and then applies then to an input image. All these transformations are parametrized with one value that roughly dictates the intensity of the transformation; this value is also chosen uniformly at random. Optimal amounts of transformations and parameter ranges for the data can be found by cross-validating the model, as the search space for the augmentation parameters is kept small enough to allow for fast cross-validation (Cubuk et al., 2020). Transformations used include color changes, translations, and blur. The full list of transformations available for RandAugment is available in the appendix of Sohn et al. (2020).

We denote the weakly augmented version of an image $\mathbf{x}$ with $\alpha_w(\mathbf{x})$, and the strongly augmented version of the same image with $\alpha_s(\mathbf{x})$. We use $\alpha(\mathbf{x})$ to denote a generic augmented image, where the augmentation can be either weak or strong.

Representation model Second, we build a CNN model to learn informative representations of input data. We use the popular ResNet50 CNN architecture (He et al., 2016), which has been shown to work well in consistency-regularized SSL applications (Tarvainen and Valpola, 2017; Berthelot et al., 2019b; Sohn et al., 2020), and has previously been used for the analysis of protest images (Won et al., 2017; Torres and Cantú, 2020). The model will take as input either a weakly or strongly augmented image, and produce a lower-dimensional representation of it.

ResNet50 consists of 50 convolutional layers with ReLU activations and maxpooling layers stacked in between. The main idea behind the architecture is that of residual learning: a layer receives as input not only the output of its predecessor, but also the output of a parent layer a few steps prior. This has been shown to lead to models that are more easily optimized and learn more

effective representations of their inputs. The final layer produces a $7 \times 7 \times 2048$ feature map, which we average pool over the first two dimension, producing a 2048-long vector representation of the input image. We denote this representation with $h(\mathbf{x})$ for an image or $h(\alpha(\mathbf{x}))$ for an augmented image.

Classification head After constructing a representation of the input data, we feed it to a small classification head. Following Chen et al. (2020b), we use a three-layer NN, where the final three layers are all fully-connected and increasingly smaller in the number of nodes, and with ReLU activations in between. We denote the prediction output by this classification head as $g(\mathbf{x}) = \sigma(\mathbf{W}_3 ReLU(\mathbf{W}_2 ReLU(\mathbf{W}_1(h(\mathbf{x})))))$, where $\mathbf{W}_l$ is a matrix of weights corresponding to layer l, and σ is a softmax function mapping to the $[0, 1]$ space, like the sigmoid function.

Loss function The key element of our framework is a loss function that combines supervised learning on the labeled data with consistency regularization and self-training on the unsupervised data. We start with the supervised component. On the supervised portion of the data, we use a standard categorical cross-entropy loss, which is defined as follows:

$$\ell^l = -\frac{1}{n^l} \sum_{i=1}^{n^l} \sum_{c=1}^{C} y_i^l(c) \log(g(\alpha_w(\mathbf{x}_i^l))). \tag{3}$$

Note that labeled images are weakly augmented before being used with the model. This is standard practice in supervised learning (Perez and Wang, 2017) and leads to good performance in our application.

First, we implement consistency regularization into our loss. For the unsupervised portion of our model we use a contrastive loss known as *Nt-Xent*: a normalized, temperature-scaled, cross-entropy loss (Sohn, 2016; Chen et al., 2020a). This loss aims at enforcing similarity between the learned representation for strongly and weakly augmented versions of the same image. Let the normalized cosine similarity between two vectors $\mathbf{a}$ and $\mathbf{b}$ be defined as: $\text{sim}(\mathbf{a}, \mathbf{b}) = \frac{\mathbf{a}^T \mathbf{b}}{\|\mathbf{a}\| \|\mathbf{b}\|}$, where $\|\mathbf{a}\|$ is the L_2 norm of $\mathbf{a}$. In addition, let τ be a scalar temperature hyperparameter. The

Nt-Xent loss for a weakly augmented unlabeled image $\mathbf{x}_i$ is defined as:

$$\ell_i^w = -\log \frac{\exp(\text{sim}(h(\alpha_w(\mathbf{x}_i^u)), h(\alpha_s(\mathbf{x}_i^u)))/\tau)}{\sum_{k=1}^{n^u} \exp(\text{sim}(h(\alpha_w(\mathbf{x}_i^u)), h(\alpha_s(\mathbf{x}_k^u)))/\tau) + \exp(\text{sim}(h(\alpha_w(\mathbf{x}_i^u)), h(\alpha_w(\mathbf{x}_k^u)))/\tau)}, \tag{4}$$

and the Nt-Xent for a strongly augmented image is defined in a similar way:

$$\ell_i^s = -\log \frac{\exp(\text{sim}(h(\alpha_w(\mathbf{x}_i^u)), h(\alpha_s(\mathbf{x}_i^u)))/\tau)}{\sum_{k=1}^{n^u} \exp(\text{sim}(h(\alpha_s(\mathbf{x}_i^u)), h(\alpha_s(\mathbf{x}_k^u)))/\tau) + \exp(\text{sim}(h(\alpha_s(\mathbf{x}_i^u)), h(\alpha_w(\mathbf{x}_k^u)))/\tau)}. \tag{5}$$

Note that this loss aims at penalizing dissimilarity between weak and strong augmentations of the same image (note the minus sign), while rewarding dissimilarity between weak/strong augmented versions of $\mathbf{x}_i$ and every other image in the dataset.

To implement a soft self-training component, we weight the cumulative Nt-Xent for each example by the max predicted probability for each class. This is a soft version of self-training, in which hard labels are assigned to examples for which predicted probability is above a certain threshold (usually between .7 and .9) (Lee, 2013). Let $q_i^u = \max_{c=1,\ldots,n_c} g(\alpha_w(\mathbf{x}_i^u))$ represent the largest predicted probability for a class for input image $\mathbf{x}_i$. We use this probability to weight the Nt-Xent loss as follows:

$$\ell_i^u = \frac{q_i^u(\ell_i^w + \ell_i^s)}{\sum_{k=1}^{n^u} q_k}. \tag{6}$$

In this way, inputs that the model is more confident on will get more weight in the regularization component of the loss. This will, in turn, induce better representations for those latent components of the visual data that are informative about predictions: if the model is very confident in which class an image belongs to, and if the strongly and weakly augmented images have similar representations, then we can conclude that the model has been successful in encoding only those graphical elements that are related with the outcome, and has managed to discard graphical alterations, such

as those induced by the transformations applied to strongly augmented images.

The final loss is given by a weighted linear combination of the two losses: $\ell = \ell^l + \lambda \ell^u$, where λ is a tunable trade-off hyperparameter, and $\ell^u = \sum_{i=1}^{n^u} \ell_i^u$. This loss improves over traditional supervised losses for the same problem by taking advantage of both the labeled data, and in the natural separation between classes in the unlabeled data.

Loss for multi-label prediction In our application to detecting the visual components of protest, as well as in other practical cases, researchers might be interested in detecting the presence/absence of certain visual elements in their images: this implies that a single image could belong to multiple classes. To adjust our proposed methodology to this setting, we redefine our labels as being a binary vector that doesn't necessarily have to be one-hot encoded, i.e., multiple entries can be one. Under this definition all the quantities in the loss can remain as they are, except for the pseudo-label weights. In this case it's not enough to just take the max over all classes, as all classes should be assigned some weight, since they can all be present at the same time in an image. Because of this we define multiclas weights as:

$$r_i^u = \sum_{c=1}^{n_c} \max(g(\alpha_w(\mathbf{x}_i^u)), 1 - g(\alpha_w(\mathbf{x}_i^u))).$$

Here we cycle over each element of interest, and record the largest probability output by our model for presence or absence of that element. The resulting probabilities are summed over all elements. The resulting weights give more importance to examples for which the model outputs confident predictions for many elements.

3.1 Training and Optimization

Optimization We adopt all the common best practices of SSL when training our model. SGD optimizer, as (Sohn et al., 2020) find that other optimization methods underperform in a setting similar to ours. Because of this, loss quantities defined on the whole data in the previous section

are in practice computed on mini-batches of data of size m^u for the unlabeled data and m^l for the labeled data. We use an exponential moving average of the weights, as in (Tarvainen and Valpola, 2017), coupled with a weight decay parameter, denoted by β. We gradually shift weight towards the unlabeled portion of the loss as training proceed, linearly increasing λ, as recommended in Lee (2013). Even though Sohn et al. (2020) find that this is unnecessary for their model, we find that it improves performance in our context. Learning rate is gradually decayed according to a cosine decay schedule (Loshchilov and Hutter, 2016), where, if the initial learning rate is η, then the learning rate at step t of T total is given by $\eta \cos\left(\frac{7\pi t}{16T}\right)$. Values for all these hyperparameters are given in our application.

Training on freely available resources One important part of our framework is that our model should be trainable with low-cost or free computational resources. Modern CNN methods often require large pools of computational resources that vastly exceed what is available to junior or independent researchers. Because of this, our implementation relies exclusively on open-source python packages and on the computational resources provided for free in Google Colaboratory.[1] Using the freely available Cloud Tensor Processing Units (TPU) provided with Colab, our model can be trained on around 50 thousand images in about 3 hours.

Our model can be fully implemented and trained with freely available resources in a relatively short amount of time, at least compared to other existing models. Coupled with the little need for hand-labelled examples, this is pivotal to making our methodology accessible to a large group of social scientists that are potentially interested in the analysis of image data.

3.2 Relationship with Existing Work

This work is related to two areas of research: first, it is related to work on detection of protest events in visual and other social media data. First, Won et al. (2017), train fully supervised detection model on the same dataset we employ in this work, using all the labels available in the

[1] https://research.google.com/colaboratory.

data. Their result is useful in that it provides a benchmark for our models in terms of out-of-sample performance, but their method requires a large amount of labelled data, which is costly to acquire, as explained previously. Zhang and Pan (2019) focus on the problem of identifying collective action events from social media posts, but their approach is different from ours in that it also employs textual data, in addition to images, and it uses fully supervised classification to train an image model, with over 200,000 labelled examples. Finally, Torres (2020) use a fully unsupervised methodology to extract visual elements from images of protests, however this approach only offers an approximation of the presence/absence of the desired elements, as it relies on attributing meaning to clusters of graphical elements after they are constructed.

Second, this work is related to the general area of semi-supervised image classification in deep learning. We use insights from several sources in constructing our algorithm (Tarvainen and Valpola, 2017; Berthelot et al., 2019b,a; Sohn et al., 2020; Chen et al., 2020a; Zhang et al., 2016). These algorithms make use of different loss functions than ours. We focus on comparing performance of our method to the approaches of (Sohn et al., 2020; Chen et al., 2020b), as these are the most recent and best performing models for SSL. We note that these models are trained and tested on generic datasets of labeled images, such as imagenet or SVHN, and our aim is not to improve on these generic methods, but to offer a tool specifically tailored to social-science data.

4 The Visual Elements of Protest

We now validate our proposed method by applying it to two related tasks: first, we detect images of protests and public rallies among images shared on social media. Second, we detect which images of protest contain each of ten different visual elements related to violence. These elements are listed in Table 1. This is an important measurement and data collection task for political science, as direct analysis of social media images permits a better understanding of how the public and the participants themselves frame these events (Won et al., 2017).

Component	Description
Sign	A protester holding a visual sign (on paper, panel, or wood).
Photo	A protester holding a sign containing a photograph of a person (politicians or celebrities)
Fire	There is fire or smoke in the scene.
Law enf.	Police or troops are present in the scene.
Children	Children are in the scene.
Group 20	There are roughly more than 20 people in the scene.
Group 100	There are roughly more than 100 people in the scene.
Flag	There are flags in the scene
Night	It is at night.
Shout	One or more people shouting.

Table 1: Visual components of protest images. Reproduced from Won et al. (2017).

Data We employ the UCLA protest dataset, originally collected by Won et al. (2017). The dataset contains 40,674 images originally shared on social media, of which 11,659 depict protest events. The dataset was created to purposefully include hard negatives that would be difficult to distinguish from protest, such as pictures of concert or stadium crowds. Crucially, the dataset was hand-annotated by human coders through Amazon MTurk: this will allow us to compare the performance of our model to a labelled benchmark. In addition to this, Won et al. (2017) train a fully-supervised CNN on all the images in the dataset.

The data was preprocessed by resizing the images to squares of side 224 pixels, and by standardizing every pixel to be between 0 and 1.

Comparison We compare to two models: first, a ResNet50 trained on fully supervised imagenet, and then fine-tuned the same subset of labelled images as our method. Second, we compare to Fixmatch, the semi-supervised approach of Sohn et al. (2020). This model has been shown to perform well on several semi-supervised benchmark dataset and represents a good point of comparison for our method.

Results Results for the task of classifying which social media images depict protest are reported in Table 2. The data is heavily imbalanced in favor of negatives, so we focus on interpreting AUC figures as opposed to raw accuracy. Our approach performs better than all other approaches it is compared to in this task. Interestingly, going from 100 to 1000 labels only brings about around a

	AUC			ACC		
N. labeled	Simple	Fixmatch	This paper	Simple	Fixmatch	This paper
100	0.76	0.754	0.842	0.759	0.76	0.745
200	0.781	0.797	0.855	0.655	0.75	0.747
500	0.833	0.865	0.884	0.618	0.785	0.791
1000	0.849	0.887	0.92	0.758	0.737	0.812

Table 2: Results on detecting images of protests from a larger set. The left panel reports Area Under the Curve (AUC) for three models, while the right side reports raw Accuracy (ACC). There are 40,674 images in the data, 11,659 of which depict protest events.

8% increase in prediction accuracy in our model. This could be evidence of the fact that our model is already learning useful clusters of input images in feature space.

Turning to the task of detecting visual components of protest images, the advantages of our approach over a simple fine-tuned model are clear: our model performs better than the simple model for almost all components and almost all amounts of labels. Comparison with Fixmatch is less simple: our approach seems to do better at labelling certain components, while Fixmatch seems to perform better when labelling others. Interestingly, we sometimes observe slight increases in performance for certain components when the number of labels decreases. This could be due to the model converging on a slightly different optimum that favors certain components over others with fewer labels. Clearly, certain components are easier to detect than others: fire, presence of police and large groups are visually easier to distinguish than children or the presence of photographs within signs. This is for clear reasons: the former bear distinguishing visual elements such as shapes or colors, while the latter don't have any clear visual distinguishing features, at least at this scale.

Overall, our proposed method performs well for this task, and we show that it can reliably be used to detect images of protest among social media posts, as well as several visual components among them.

N. labeled	500			200			100			50		
Component	Simple	Fixmatch	This paper	Simple	Fixmatch	This paper	Simple	Fixmatch	This paper			
Sign	0.592	0.228	0.527	0.591	0.228	0.529	0.475	0.254	0.548	0.487	0.259	0.545
Photo	0.484	0.733	0.841	0.465	0.696	0.787	0.513	0.658	0.733	0.365	0.687	0.618
Fire	0.929	0.979	0.923	0.887	0.944	0.88	0.904	0.933	0.871	0.863	0.943	0.869
Police	0.712	0.906	0.951	0.836	0.877	0.902	0.862	0.83	0.872	0.614	0.833	0.776
Children	0.806	0.817	0.828	0.648	0.704	0.759	0.711	0.737	0.628	0.733	0.715	0.545
Group > 20	0.718	0.33	0.862	0.752	0.289	0.8	0.651	0.298	0.77	0.667	0.342	0.729
Group > 100	0.91	0.947	0.931	0.901	0.938	0.932	0.878	0.947	0.889	0.791	0.951	0.88
Flag	0.799	0.824	0.735	0.805	0.805	0.696	0.79	0.782	0.612	0.749	0.815	0.568
Night	0.725	0.856	0.801	0.696	0.866	0.765	0.644	0.856	0.767	0.581	0.842	0.638
Shouting	0.69	0.781	0.875	0.611	0.772	0.833	0.555	0.773	0.795	0.576	0.656	0.708

Table 3: AUC Results on detecting components of protests in protest images. There are 11, 659 images of protest, and only a small amount that contain each component. We only report AUC figures for this task as the data is heavily imbalanced.

5 Conclusion

In this paper, we have proposed a methodology for automatically labelling a large set of images needing only a small amount of human-labelled examples. Our methodology is explicitly intended for lowering the cost of image data analysis for a large group of social scientists. We have employed recent advances in deep-learning and semi-supervised learning to define a CNN model able to leverage both the unlabeled and labeled data to efficiently annotate the entire dataset. In our application, we have shown that our methodology performs well for the task of detecting images of protest among larger sets of social media images. This is, in turn, important for understanding how the public and the participants active on social media frame movements.

There are several possible extensions for our methodology: first, it is possible to adapt the consistency regularization framework we propose to the problem of object detection within images, by training not only on presence/absence of elements within images, but also on the location of these elements within the images. Second, it is possible to extend the consistency regularization framework specifically to the multi-label classification problem, by forcing inputs with the same class to be close only in certain parts of the learned representation space.

In conclusion, our methodology makes advancements towards enabling a larger number of researchers to analyze image data independently of the resources available to them, therefore offering a new and powerful way to measure and identify important concepts in political science.

References

Massih-Reza Amini and Patrick Gallinari. Semi-supervised logistic regression. In ECAI, pages 390–394, 2002.

L Jason Anastasopoulos, Dhruvil Badani, Crystal Lee, Shiry Ginosar, and Jake Williams. Photographic home styles in congress: a computer vision approach. arXiv preprint arXiv:1611.09942, 2016.

Pablo Barberá, Andreu Casas, Jonathan Nagler, Patrick J Egan, Richard Bonneau, John T Jost, and Joshua A Tucker. Who leads? who follows? measuring issue attention and agenda setting by legislators and the mass public using social media data. American Political Science Review, 113 (4):883–901, 2019.

David Berthelot, Nicholas Carlini, Ekin D Cubuk, Alex Kurakin, Kihyuk Sohn, Han Zhang, and Colin Raffel. Remixmatch: Semi-supervised learning with distribution matching and augmentation anchoring. In International Conference on Learning Representations, 2019a.

David Berthelot, Nicholas Carlini, Ian Goodfellow, Nicolas Papernot, Avital Oliver, and Colin A Raffel. Mixmatch: A holistic approach to semi-supervised learning. In Advances in Neural Information Processing Systems, pages 5049–5059, 2019b.

Andreu Casas and Nora Webb Williams. Images that matter: Online protests and the mobilizing role of pictures. Political Research Quarterly, 72(2):360–375, 2019.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. arXiv preprint arXiv:2002.05709, 2020a.

Ting Chen, Simon Kornblith, Kevin Swersky, Mohammad Norouzi, and Geoffrey Hinton. Big self-supervised models are strong semi-supervised learners. arXiv preprint arXiv:2006.10029, 2020b.

Ekin D Cubuk, Barret Zoph, Dandelion Mane, Vijay Vasudevan, and Quoc V Le. Autoaugment: Learning augmentation strategies from data. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 113–123, 2019.

Ekin D Cubuk, Barret Zoph, Jonathon Shlens, and Quoc V Le. Randaugment: Practical automated data augmentation with a reduced search space. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops, pages 702–703, 2020.

James N Druckman and Michael Parkin. The impact of media bias: How editorial slant affects voters. The Journal of Politics, 67(4):1030–1049, 2005.

Timnit Gebru, Jonathan Krause, Yilun Wang, Duyun Chen, Jia Deng, Erez Lieberman Aiden, and Li Fei-Fei. Using deep learning and google street view to estimate the demographic makeup of neighborhoods across the united states. Proceedings of the National Academy of Sciences, 114 (50):13108–13113, 2017.

Franklin D Gilliam Jr and Shanto Iyengar. Prime suspects: The influence of local television news on the viewing public. American Journal of Political Science, pages 560–573, 2000.

Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 580–587, 2014.

Maria Elizabeth Grabe and Erik Page Bucy. Image bite politics: News and the visual framing of elections. Oxford University Press, 2009.

Yves Grandvalet and Yoshua Bengio. Semi-supervised learning by entropy minimization. In Advances in neural information processing systems, pages 529–536, 2005.

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

Jens Hainmueller and Dominik Hangartner. Who gets a swiss passport? a natural experiment in immigrant discrimination. American political science review, pages 159–187, 2013.

Lene Hansen. How images make world politics: International icons and the case of abu ghraib. Review of International Studies, 41(2):263–288, 2015.

Danny Hayes, Jennifer L Lawless, and Gail Baitinger. Who cares what they wear? media, gender, and the influence of candidate appearance. Social Science Quarterly, 95(5):1194–1212, 2014.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 770–778, 2016.

Irina Higgins, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. beta-vae: Learning basic visual concepts with a constrained variational framework. Iclr, 2(5):6, 2017.

Yusaku Horiuchi, Tadashi Komatsu, and Fumio Nakaya. Should candidates smile to win elections? an application of automated face recognition technology. Political Psychology, 33(6):925–933, 2012.

Jungseock Joo and Zachary C Steinert-Threlkeld. Image as data: Automated visual content analysis for political science. arXiv preprint arXiv:1810.01544, 2018.

Jungseock Joo, Weixin Li, Francis F Steen, and Song-Chun Zhu. Visual persuasion: Inferring communicative intents of images. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 216–223, 2014.

Jungseock Joo, Francis F Steen, and Song-Chun Zhu. Automated facial trait judgment and election outcome prediction: Social dimensions of face. In Proceedings of the IEEE international conference on computer vision, pages 3712–3720, 2015.

Diederik P Kingma and Max Welling. Auto-encoding variational bayes. arXiv preprint arXiv:1312.6114, 2013.

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

Chappell Lawson and James A McCann. Television news, mexico's 2000 elections and media effects in emerging democracies. British Journal of political science, 35(1):1–30, 2005.

Chappell Lawson, Gabriel S Lenz, Andy Baker, and Michael Myers. Looking like a winner: Candidate appearance and electoral success in new democracies. World Politics, 62(4):561–593, 2010.

Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. Backpropagation applied to handwritten zip code recognition. Neural computation, 1(4):541–551, 1989.

Yann LeCun, Yoshua Bengio, et al. Convolutional networks for images, speech, and time series. The handbook of brain theory and neural networks, 3361(10):1995, 1995.

Dong-Hyun Lee. Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks. In Workshop on challenges in representation learning, ICML, volume 3, 2013.

Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 2117–2125, 2017.

Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. arXiv preprint arXiv:1608.03983, 2016.

David A Makin, Dale W Willits, Wendy Koslicki, Rachael Brooks, Bryce J Dietrich, and Rachel L Bailey. Contextual determinants of observed negative emotional states in police–community interactions. Criminal justice and behavior, 46(2):301–318, 2019.

Kyle Mattes and Caitlin Milazzo. Pretty faces, marginal races: Predicting election outcomes using trait assessments of british parliamentary candidates. Electoral Studies, 34:177–189, 2014.

Caitlin Milazzo and Kyle Mattes. Looking good for election day: Does attractiveness predict electoral success in britain? The British Journal of Politics and International Relations, 18(1): 161–178, 2016.

Saffron J O'Neill and Nicholas Smith. Climate change and visual imagery. Wiley Interdisciplinary Reviews: Climate Change, 5(1):73–87, 2014.

Luis Perez and Jason Wang. The effectiveness of data augmentation in image classification using deep learning. arXiv preprint arXiv:1712.04621, 2017.

Thomas E Powell, Hajo G Boomgaarden, Knut De Swert, and Claes H de Vreese. A clearer picture: The contribution of visuals and text to framing effects. Journal of communication, 65 (6):997–1017, 2015.

Ludovic Rheault and Sophie Borwein. Multimodal techniques for the study of a ect in political videos. Working Paper, 2019.

Margaret E Roberts, Brandon M Stewart, and Edoardo M Airoldi. A model of text for experimentation in the social sciences. Journal of the American Statistical Association, 111(515):988–1003, 2016.

Emma Rodman. A timely intervention: Tracking the changing meanings of political concepts with word vectors. Political Analysis, 28(1):87–111, 2020.

Mehdi Sajjadi, Mehran Javanmardi, and Tolga Tasdizen. Regularization with stochastic transformations and perturbations for deep semi-supervised learning. In Advances in neural information processing systems, pages 1163–1171, 2016.

Maya Sen and Omar Wasow. Race as a bundle of sticks: Designs that estimate effects of seemingly immutable characteristics. Annual Review of Political Science, 19, 2016.

Connor Shorten and Taghi M Khoshgoftaar. A survey on image data augmentation for deep learning. Journal of Big Data, 6(1):60, 2019.

Kihyuk Sohn. Improved deep metric learning with multi-class n-pair loss objective. In Advances in neural information processing systems, pages 1857–1865, 2016.

Kihyuk Sohn, David Berthelot, Chun-Liang Li, Zizhao Zhang, Nicholas Carlini, Ekin D Cubuk, Alex Kurakin, Han Zhang, and Colin Raffel. Fixmatch: Simplifying semi-supervised learning with consistency and confidence. arXiv preprint arXiv:2001.07685, 2020.

Denis G Sullivan and Roger D Masters. " happy warriors": Leaders' facial displays, viewers' emotions, and political support. American Journal of Political Science, pages 345–368, 1988.

Antti Tarvainen and Harri Valpola. Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results. In Advances in neural information processing systems, pages 1195–1204, 2017.

Dustin Tingley. Face-off: Facial features and strategic choice. Political Psychology, 35(1):35–55, 2014.

Alexander Todorov, Anesu N Mandisodza, Amir Goren, and Crystal C Hall. Inferences of competence from faces predict election outcomes. Science, 308(5728):1623–1626, 2005.

Michelle Torres. Give me the full picture: Using computer vision to understand visual frames and political communication. Working paper, 2020.

Michelle Torres and Francisco Cantú. Learning to see: Convolutional neural networks for the analysis of social science data. Working paper, 2020. URL https://franciscocantu.github.io/Papers/PA_MTFC_LearningToSee_Rev01.pdf.

Jason Wang, Luis Perez, et al. The effectiveness of data augmentation in image classification using deep learning. Convolutional Neural Networks Vis. Recognit, 11, 2017.

Nils B Weidmann and Sebastian Schutte. Using night light emissions for the prediction of local wealth. Journal of Peace Research, 54(2):125–140, 2017.

Nora Webb Williams, Andreu Casas, and John D Wilkerson. Images as data for social science research: An introduction to convolutional neural nets for image classification. Elements in Quantitative and Computational Methods for the Social Sciences, 2020.

Donghyeon Won, Zachary C Steinert-Threlkeld, and Jungseock Joo. Protest activity detection and perceived violence estimation from social media images. In Proceedings of the 25th ACM international conference on Multimedia, pages 786–794, 2017.

Nan Xi, Di Ma, Marcus Liou, Zachary C Steinert-Threlkeld, Jason Anastasopoulos, and Jungseock Joo. Understanding the political ideology of legislators from social media images. In Proceedings of the International AAAI Conference on Web and Social Media, volume 14, pages 726–737, 2020.

Qizhe Xie, Zihang Dai, Eduard Hovy, Minh-Thang Luong, and Quoc V Le. Unsupervised data augmentation for consistency training. arXiv preprint arXiv:1904.12848, 2019.

Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. arXiv preprint arXiv:1611.03530, 2016.

Han Zhang and Jennifer Pan. Casm: A deep-learning approach for identifying collective action events with text and image data from social media. Sociological Methodology, 49(1):1–57, 2019.

Xiaojin Zhu and Andrew B Goldberg. Introduction to semi-supervised learning. Synthesis lectures on artificial intelligence and machine learning, 3(1):1–130, 2009.