An Introduction to Neural Networks for the Social Sciences[*]

Gechun Lin [†] Christopher Lucas [‡]

Forthcoming in *The Oxford Handbook of Methodological Pluralism in Political Science*

Abstract

Social scientists now regularly employ computational models, most often to create proxy variables for theoretical concepts that are difficult to observe directly. In this chapter, we discuss how the application of these models differs from traditional quantitative methods common to the social sciences. We then introduce a particular method — neural networks — by building from a model familiar to social scientists: logistic regression. To build intuition about how neural networks flexibly approximate complex functions, we compare the performance of logistic regression to neural networks in a simple simulation. We then further demonstrate the flexibility of neural networks by introducing convolutional and recurrent neural networks and their application to image and text classification, respectively. Throughout the chapter, we relate approaches to machine learning with neural networks to common statistical practices in social science. We conclude with a discussion of new methodological challenges posed by reliance on neural networks and open areas of research.

Keywords: Machine learning, computational social science, neural networks

[*]For helpful comments and discussions, we thank Dino Christenson, Connor Huff, Gary King, Dean Knox, Caleb Lucas, Jacob Montgomery, Dustin Tingley, Michelle Torres, and Xiang Zhou.

[†]Washington University in St. Louis, lingechun@wustl.edu

[‡]Washington University in St. Louis, christopher.lucas@wustl.edu

1 Introduction

The rate of progress in machine learning research has accelerated dramatically due to increases in computing power, highlighted by the recent success of neural networks in previously intractable problems (LeCun et al., 2015). And while neural networks were theoretically understood decades ago, their high computational demands meant they were not practically feasible until recently, establishing new benchmarks in machine learning with text, audio, and image data.

At the same time, social scientists increasingly leverage machine learning to estimate proxy variables of latent concepts that are difficult to observe directly. In a recent article, Knox et al. (2022) reviewed papers published between 2018 and 2020 in the *American Journal of Political Science*, the *American Political Science Review*, and the *Journal of Politics*, and identified studies that employed machine learning. In the majority of these cases (approximately 68%), machine learning was utilised first to estimate a proxy variable, which was then used in a second-stage empirical test of the primary research question. However, these two developments — the recent success of neural networks and the increased use of computational methods by social scientists — have largely occurred independently of each other. For example, of the 48 papers identified by Knox et al. (2022) using machine learning, only one study (Cantú, 2019) utilized a neural network.

While there are many possible explanations for this relatively slow uptake, we point to three. First, the statistical foundations of traditional approaches to machine learning (e.g., penalized regression) are better understood than those of neural networks. As a consequence, it is easier to correct bias resulting from reliance on a proxy variable, which is a noisy approximation of the true variable of interest. Second, while computational processing power has in some sense caught up with the demands of neural networks, they still require significantly more computational resources compared to alternate models. Last, neural networks often require significantly more training data than many alternative approaches. Of these three, there is reason to suspect that at least the first two are not permanent limitations. Much work is underway to better characterize uncertainty in neural networks,[1] and the limits of computational processing continue to extend. Furthermore, while the quantity of training data required to

[1]See Gawlikowski et al. (2021) and Abdar et al. (2021) for a review of recent developments in this area.

successfully train neural networks is considerable, it can be reduced. For example, transfer learning — using a model trained on a separate task as a starting point for a new one — can considerably reduce the amount of training data required. Despite these limitations, neural networks are already in use for text and image classification in the social sciences (Torres and Cantú, 2022; Chang and Masterson, 2020; Rheault and Cochrane, 2020; Rodriguez and Spirling, 2022; Lin, 2022b), and Barari et al. (2021) use neural networks to create "deepfake" videos of politicians. In addition, research as early as Beck et al. (2000) demonstrated the potential for neural networks to outperform alternative models in some social science applications.

With this is mind, in the rest of this chapter, we provide an introduction to neural networks that builds on a statistical model common to the social sciences, logistic regression. We begin by discussing the use of machine learning to estimate proxy variables, with a focus on underappreciated consequences of doing so. Next, we highlight differences between a highly predictive model capable of learning a relatively accurate proxy and a model that credibly tests a causal claim. We then formally define neural networks using notation familiar to social scientists and conduct a simple simulation to empirically demonstrate how neural networks approximate complex functions. Subsequently, we apply neural networks to two applications to unstructured data of interest to social scientists. In the first, we classify satellite imagery, and in the second, we identify hateful speech on social media. We conclude by discussing the limitations of neural networks for social science research and how future work might address these concerns.

2 Machine Learning to Estimate Proxies for Social Science Concepts

Generally, social scientific theories are claims about causal relationships (Ashworth et al., 2015; Kellstedt and Whitten, 2018), and as a result, the most common use of statistical methods by social scientists is to evaluate and test these causal hypotheses. This contrasts with many domains of applied statistics, in which the core purpose of a model is often predictive accuracy (e.g., forecasting the weather or making algorithmic recommendations to customers). However, given that predictive accuracy is generally the primary goal in machine learning, to what end is it applied in the social sciences?

Core theoretical concepts in the social sciences are often difficult or even impossible to observe directly. For example, consider the prevalence of legislator ideal points in explanations of Congressional behavior. Though the notion of ideal points is commonly invoked in theories of American and comparative politics, a researcher cannot simply look up a legislator's true ideal point next to other biographical information. Instead, methods were developed to estimate legislator ideal points, which were then used as proxies for their true ideal point in second-stage tests of theoretical claims (Poole and Rosenthal, 1985).

Similarly, social scientists now routinely use machine learning to estimate proxies across a range of substantive areas. In this two-stage process, researchers first construct a model to predict a variable that is otherwise difficult to measure, then use the predictions as a proxy variable in a second-stage regression. These proxy variables are often but not exclusively created from "unstructured" data (data that are not obviously rectangular, such as text, images, or audio), commonly using off-the-shelf statistical methods for prediction.

This practice is effectively equivalent to multiple imputation with auxiliary variables. In multiple imputation, a researcher estimates missing data from other available data, and then uses the imputed data in a second-stage test of primary hypotheses. Often, the first stage, in which missing data are imputed, leverages variables not used in the second-stage test (auxiliary variables). In this sense, a researcher estimating the ideal point of a legislator from their votes is effectively imputing the missing variable (ideal point) from auxiliary variables (legislator voting record). This equivalence highlights some of the challenges when using machine learning to estimate proxies of missing variables. For example, it is well known that ignoring the uncertainty in predicted covariates from multiple imputation can bias estimates in the second-stage regression (Blackwell et al., 2017). However, this source of bias is rarely acknowledged when using machine learning to estimate proxies.

3 Building Neural Networks from Logistic Regression

Having summarized the general use cases of machine learning in the social sciences, we now introduce one approach to machine learning: neural networks. To do so, we build from a statistical context familiar to social scientists — regression with a binary dependent variable — highlighting that while the statistical model is similar in both cases, the concerns change when the model is used to estimate a proxy rather than a causal effect. At the outset of this discussion, we note that our introduction to neural networks is necessarily narrow. We focus on one way machine learning is transforming the social sciences and introduce a particular approach (neural networks) through analogy to methods familiar to social scientists. For a deeper treatment of neural networks, we point readers to Goodfellow et al. (2016).

3.1 Effect Estimation or Prediction?

Models for binary outcomes are among the first covered in introductions to statistical inference and are pervasive in the social sciences. However, many of the same models appear in similarly foundational courses in machine learning, albeit with differing motivations. Logistic regression, for example, is ubiquitous on introductory machine learning syllabi, though most social scientists would not associate logistic regression with "machine learning." For this reason, logistic regression is a useful example in this chapter, and also conveniently serves as one of the building blocks of neural networks.

Specifically, consider a binary dependent variable Y_i and a single binary independent variable D_i. For the time being, assume that D_i is randomly assigned, so there is no need to adjust for confounding variables. In this case, to estimate the causal effect of D_i on Y_i, a researcher might employ a linear probability model, where $P(Y_i = 1|D_i = d_i) = \beta_0 + d_i\beta_1$, in which the probability that the outcome Y_i equals 1 is a linear function of the explanatory variables. Though the linear probability model is often maligned for its potential to predict probabilities outside the $[0, 1]$ interval, in this case, such concerns are unwarranted. Because the model is saturated (a separate parameter exists for every permutation of possible values that the regressors might take), the linear probability model cannot generate predictions that are out-of-bounds

(Gomila, 2021). Additionally, while the errors in a linear probability model are necessarily heteroskedstic (biasing conventional estimators), heteroskedasticity-robust standard errors permit simple correction.

As a consequence, if the quantity of interest is the average effect of D_i on Y_i, in this case, the estimated coefficient on D_i provides a directly interpretable, unbiased estimate of that quantity. Alternatively, imagine using logistic regression in this context. The estimated coefficient would be the less interpretable change in log-odds of the outcome (although much can be done to make estimates more interpretable, e.g., King et al. (2000)). Moreover, a primary benefit of logistic regression — avoiding the possibility of implausible predictions — is irrelevant in this case, because the model is saturated.

Instead, consider a different case in which the target quantity of interest is actually the best prediction of the outcome given the inputs, as is generally the case when estimating a proxy. In this context, properties of the estimated coefficients are considerably less important. Rather, the quantity of interest shifts to the estimated probabilities $\hat{Y}_i$. In this context, our evaluation of the tradeoff between interpretable coefficients and possibly absurd estimates of $\hat{Y}_i$ shifts. If we are not particularly interested in estimating interpretable coefficients, it is sensible to sacrifice interpretability for a better model of the dependent variable, particularly in non-saturated cases where a linear probability model might make predictions outside the $[0, 1]$ interval. With this in mind, imagine that instead of working with a single binary predictor D_i, we have a single continuous predictor x_i. Using logistic regression, our model would take the familiar form of that shown in Equation 1. In this case, the model specification — $\beta_0 + x_i\beta_1$ — of the linear predictor (the quantity passed to the link function) remains unchanged, but the relationship between the parameters and the outcome changes due to the imposition of a non-linear link.

$$P(Y_i = 1 | X_i = x_i) = \frac{1}{1 + e^{-(\beta_0 + x_i\beta_1)}} \tag{1}$$

A saturated linear probability model is not necessarily a poor choice in the context of prediction; $\hat{Y}_i$ will simply be the mean of the outcome within the set of observations with identical values of the regressors. However, model specifications that are not saturated may generate unreasonable probabilities, and saturated models are not possible with continuous predictors. Moreover, saturated models with many discrete predictors (e.g., fixed effects) may be inefficient. In such cases, where the quantity of interest is in fact the predicted value of the outcome, a non-linear transformation — like logistic regression — is likely preferable.

As an illustration, imagine that the correct specification of the linear predictor is $\beta_0 + \beta_1 * x_i + \beta_2 * x_i^2 + \beta_3 * x_i^3 + \beta_4 * x_i^4 + \beta_5 * x_i^5$ (a polynomial of degree five). Though this is still a function of a single variable, a linear probability model will make out-of-bounds predictions under many plausible data generating processes. In such a case, a researcher with a great deal of domain-specific knowledge might in fact be able to correctly specify a model with a fifth-degree polynomial. However, in the context of prediction from unstructured data, it is often unrealistic to expect that the model is correctly specified. For example, consider text classification. In Section 4.2, we train a neural network to identify hate speech in social media posts based on the words in each post. To do so with logistic regression, a researcher might regress the label — a binary indicator for whether or not the post is an example of hate speech — on the count of terms in each post. But what is the correct model specification? Which interactions should we include? For example, the word "animal" is likely not very predictive of hate speech on its own, but posts referencing a certain group of people and also containing the word "animal" are probably examples of hate speech. But which interactions should be included? Even a researcher with expertise in online hate speech would be unable to enumerate every possible interaction.

This comparison between the linear probability model and logistic regression highlights several important considerations. Chiefly, when the quantity of interest is predictive accuracy (rather than estimated effects of one variable on another), the benefits of interpretable, non-parametric models decrease relative to the benefits of more accurate modeling of the outcome. But secondarily, and particularly when working with unstructured data, we would ideally relax the presumption of what the researcher might know about the relationship between the regressors and the outcome. It is in this task that neural networks outperform traditional approaches in some applications, most notably certain applications of classification from images, audio, and text data. In fact, neural networks are able to approximate *any* function, no matter how complex (Hornik et al., 1989).[2]

[2]Though we note that neural networks are not unique in their ability to approximate any function. For example,

In the next section, we define a simple neural network and demonstrate it with a simulation.

3.2 Logistic Regression as a Special Case of a Neural Network

Recall the logistic regression model displayed in Equation 1. In logistic regression, the linear predictor $\beta_0 + x_i\beta_1$, which ranges from $-\infty$ to ∞, is mapped to the $[0, 1]$ interval via the logistic function. In the case of logistic regression, this output is the estimated probability that $Y_i = 1$ given the inputs to the function.

Neural networks function similarly, except the output on the $[0, 1]$ interval is instead passed to a subsequent logistic function. In the simplest case, the inputs are in fact passed to multiple functionally-equivalent logistic functions, the outputs of which are then passed to one separate logistic function, which then returns $P(Y = 1|X = x_i)$. Formally, with a single predictor x_i, let α_0^k and α_i^k be the intercept and slope coefficients the k^{th} logistic function which receives x_i as input. Then, let $\hat{\gamma}_i^k$ be the output from the k^{th} logistic function, which is then passed to a final logistic function with $k + 1$ parameters (one for each $\hat{\gamma}_i^k$ plus an intercept). The final logistic function is typically called the "activation function," and every preceding function is called a "hidden unit." With this notation defined, we can represent a neural network with two hidden units as follows.

$$\hat{\gamma}_i^1 = \frac{1}{1 + e^{-(\alpha_0^1 + x_i\alpha_1^1)}} \qquad \text{(First Hidden Unit)}$$

$$\hat{\gamma}_i^2 = \frac{1}{1 + e^{-(\alpha_0^2 + x_i\alpha_1^2)}} \qquad \text{(Second Hidden Unit)}$$

$$P(Y_i = 1|X_i = x_i) = \frac{1}{1 + e^{-(\beta_0 + \hat{\gamma}_i^1\beta_1 + \hat{\gamma}_i^2\beta_2)}} \qquad \text{(Activation Unit)}$$

In this neural network, note that there is no difference between the first and second hidden units. This has several implications. Most notably, unlike logistic regression, there is no single best solution. In the example above, we could swap the estimated values of the parameters between the first and second unit and generate equivalent outputs for all inputs. Though the existence of multiple solutions is ubiquitous in machine learning (Roberts et al., 2016), this

the Stone–Weierstrass approximation theorem shows that this is also true for polynomial functions, and there is a direct correspondence between polynomial regression and neural networks (Cheng et al., 2018).

symmetry limits our ability to interpret any single estimated coefficient. However, as noted earlier in this chapter, this is not necessarily a problem in the context of estimating a proxy, where we are chiefly concerned with accurate prediction.

We can extend the number of hidden units to arbitrarily many, as shown below. In doing so, we can in fact approximate any function as the number of units goes to infinity (Hornik et al., 1989). This is because each additional logistic function provides increased flexibility to the model.

$$\begin{aligned}
\hat{\gamma}_i^1 &= \frac{1}{1+e^{-(\alpha_0^1+x_i\alpha_1^1)}} && \text{(First Hidden Unit)}\\
&\vdots \\
\hat{\gamma}_i^k &= \frac{1}{1+e^{-(\alpha_0^k+x_i\alpha_1^k)}} && (k^{th}\text{ Hidden Unit})\\
P(Y_i=1|X_i=x_i) &= \frac{1}{1+e^{-(\beta_0+\hat{\gamma}_i^1\beta_1+\ldots+\hat{\gamma}_i^k\beta_k)}} && \text{(Activation Unit)}
\end{aligned}$$

Figures1 and 2 provide graphical illustrations of logistic regression and a neural network, respectively, where each node represents a logistic function.

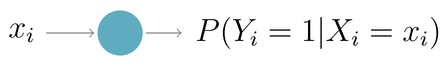


Figure 1: Logistic regression.

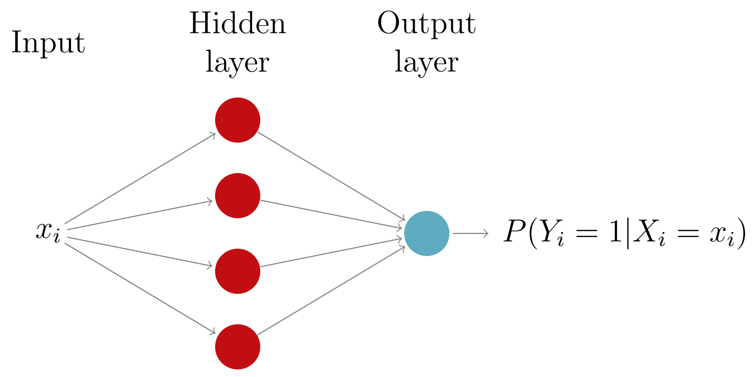


Figure 2: Neural network with one hidden layer.

3.3 A Simulation

To illustrate how neural networks can approximate complex functions, we will conduct a simple simulation. Specifically, we simulate data from a logistic regression model, where the true linear predictor is a polynomial of degree five.[3] We then sample 10,000 observations,

[3]The specification of the linear predictor used in the simulation was $f(x_i) = 0.01 + 24.54x_i - 176.7708x_i^2 + 462.3958x_i^3 - 505.7292x_i^4 + 196.3542x_i^5$. We chose this specification by first manually defining a set of points lying on a non-convex curve, then fitting a polynomial of degree five to these points. The true linear predictor used in the

drawing $x_i \sim U(0,1)$ and y_i according to the probabilities given by the preceding model.

The un-dashed, solid line in Figure 3 plots the true $P(Y_i = 1|X_i = x_i)$ over the $[0,1]$ interval. By design of the simulation, a logistic regression with the correctly specified linear predictor (a polynomial of degree five) would return an unbiased estimate of the true model. But what if the true specification of the linear predictor was not known? What if a model only received the correct inputs, but not the correct polynomials (or interactions, in the case of multiple inputs)?

In this case, if we fit a logistic regression with a linear predictor $\beta_0 + \beta_1 x_i$, predictably, the model will perform poorly. Most notably, the logistic function is monotonically increasing or decreasing in the input. By visibly inspecting the function in Figure 3, however, it is obvious that the true, latent function is not monotonic. As a result, logistic regression will provide a poor approximation of this function without correctly specifying the polynomial. However, a neural network with four hidden units can provide a decent approximation of this function without any specification of a particular functional form.

The dotted grey line in Figure 3 displays the predicted curve of a logistic function estimated without the inclusion of a polynomial (i.e., the linear predictor is $\beta_0 + \beta_1 x_i$). It is labeled "No Hidden Units," since we can think of the logistic regression as simply a neural network without any hidden units. The remaining two lines plot the approximated curve from neural networks with two and four hidden units. Note that as the number of hidden nodes increases, the neural network's ability to approximate the true, latent function improves. Specifically, note that the model with no hidden units is approximately linear in the domain of the simulation. Were we to expand the domain of the plot, we would observe that this line is simply the S-shaped curve of logistic regression. This is because without any hidden units, the neural network in this example is simply a logistic regression.

simulation is the estimated model from this exercise.

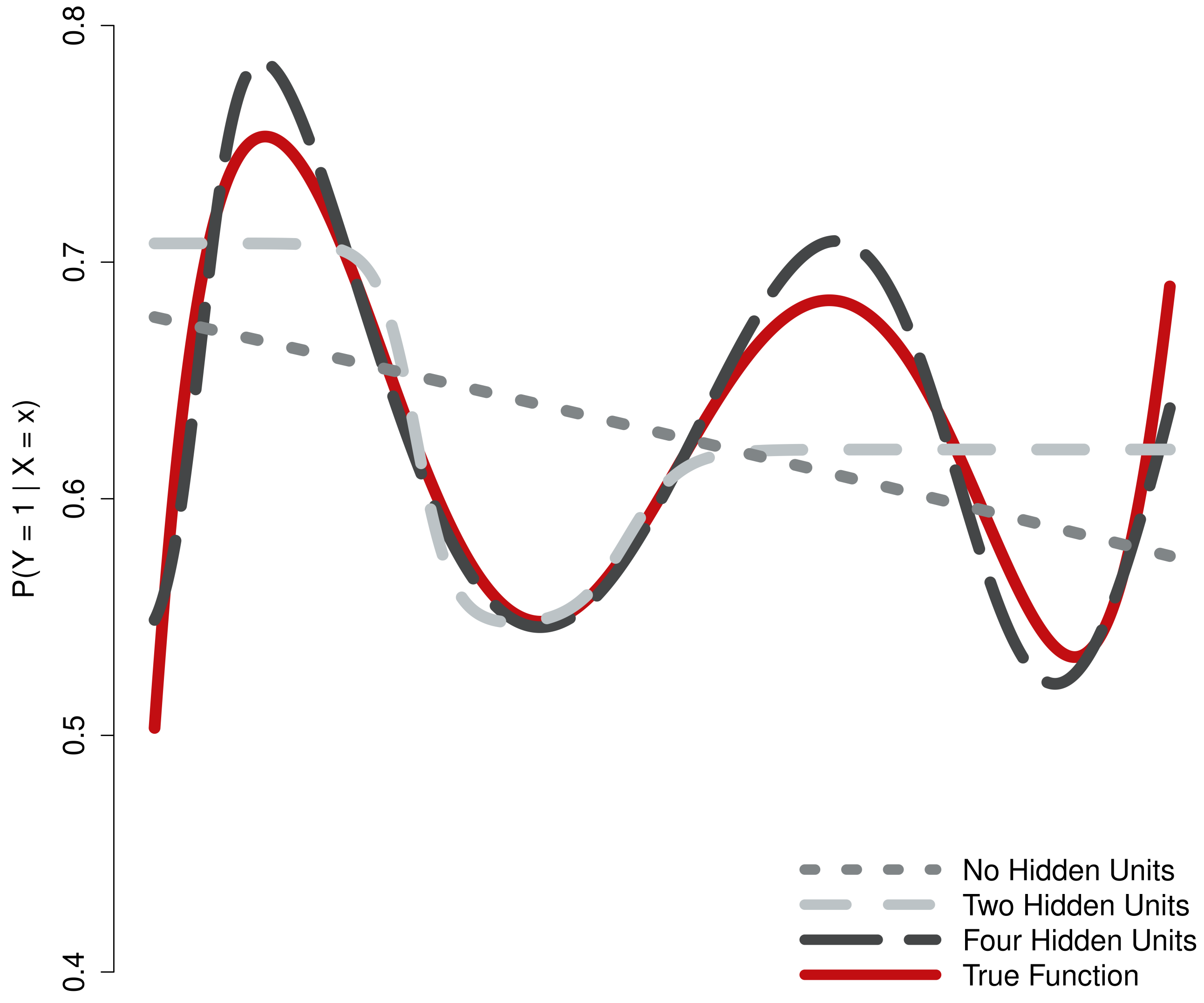


Figure 3: Results from a simulation demonstrating how hidden units in a neural network flexibly approximate more complex functions. The solid line plots the true probability that Y equals 1, given X. The dashed line labeled "No Hidden Units" plots the estimated probability that Y equals 1, given X, estimated by a logistic regression of $Y \sim X$. The remaining two dashed lines — "Two Hidden Units" and "Four Hidden Units" — plot the estimated probability that Y equals 1, given X, by neural networks with two and four hidden units, respectively.

Alternatively, note that the curves with two and four hidden units do a better job approximating the true latent curve. Specifically, the curve with two hidden units estimates a function that is non-monotonic in X, but is not sufficiently flexible to capture the local maxima of the true function. Made more flexible with the addition of two hidden units, the neural network with four hidden units is able to capture these points.

In this simple simulation, it is possible that a researcher might be able to correctly specify a polynomial of degree five in the linear predictor, in which case a logistic regression would of course be sufficient by design. In the next section, we conduct two applications in which the covariates are terms in documents and the pixel intensities of images, for which it would not be reasonable to expect that any researcher could correctly specify the model.

3.4 Additional Flexibility in Neural Networks

We have thus far explained how neural networks with a single hidden layer composed of many logistic functions can learn complex functions without the researcher knowing much about the particular functional relationship between the covariates and the dependent variable. There are additional flexibilities when constructing neural networks. For example, while it is common to use a logistic function, functions in the hidden layer can instead use the hyperbolic tangent function, which is a rescaled logistic function that instead ranges from -1 to 1. In fact, many different functions can be used in place of the logistic function. In certain applications, doing so may decrease the amount of time necessary to fit the model or prevent model fitting from terminating prematurely. Additionally, and as we briefly noted above, neural networks may be composed of many hidden layers, by feeding the output from one hidden layer to another. Moreover, units in a given hidden layer need not receive the same input. In the next section, we discuss why it is sometimes preferable to provide different inputs to different units in a hidden layer.

4 Demonstrating Neural Networks with Texts and Images

To demonstrate neural networks with real (i.e., not simulated) data, we provide two applications, which highlight how neural networks can be adapted to more complex tasks. In the previous section, we used a simulation to show how neural networks flexibly approximate complex functions without requiring the correct model specification. The neural networks that we used to do so are in fact a class of neural networks called "fully-connected" networks, in which each logistic function in the hidden layer receives every covariate in the set of predictors. In some contexts, however, it can be useful to provide a smaller set of inputs to each logistic function, especially where there is a substantive justification for narrowing the set of inputs, and particularly when a fully-connected network would require an unnecessarily large number of parameters, as is often the case with unstructured data like images and text. In the first application, we show how the structure of a neural network can be altered to classify images.

4.1 Classifying Satellite Images with a Convolutional Neural Network

Image classification illustrates why fully-connected neural networks are often not ideal. An image that is 256 pixels wide and 256 pixels tall contains 65,536 pixels. If the color intensity of each pixel was passed to a single logistic function, that function will have one parameter for every pixel. With 16 hidden units, there would be over 1,000,000 parameters. And, importantly, neighboring pixels are more informative than pixels on the opposite side of the image, so a fully-connected neural network unnecessarily adds additional parameters.

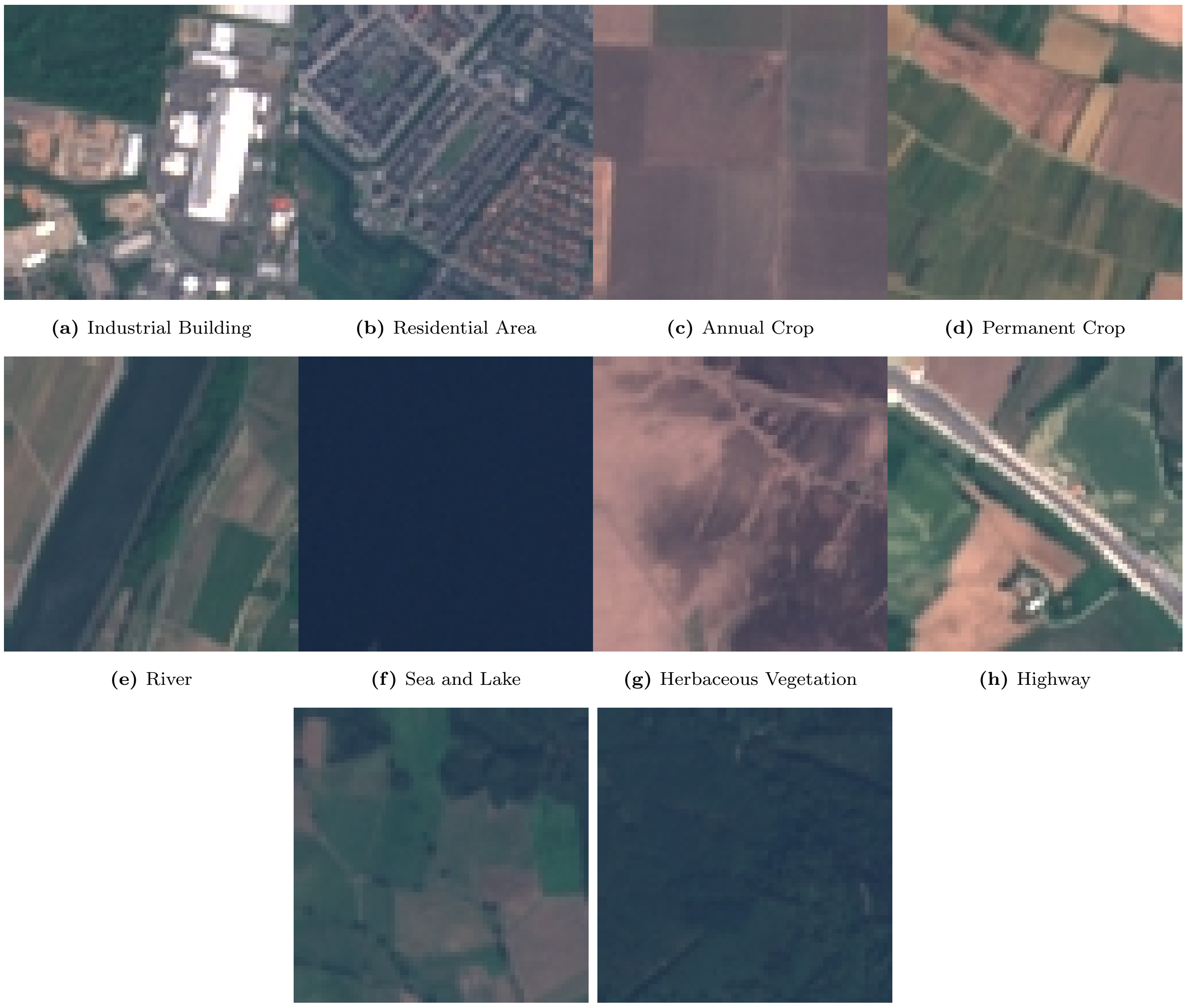
Instead, when classifying images, each logistic function in the hidden layer might receive as input the intensity of small grid of pixels (2x2, 3x3, etc), flattened into a vector. Neural networks of this sort are called convolutional neural networks.[4] This is in fact similar to some approaches to spatial data analysis in the social sciences. Spatial autocorrelation — dependence between geographically proximate units — is a core concern in most spatial analyses. To correct for autocorrelation, it is common to include a spatial lag, which is simply the weighted average of neighboring values. A convolutional neural network accomplishes precisely this, except the value of the unit itself is also included in the weighted average calculated by a convolutional neural network. In spatial statistics, this is commonly called a window average. In other words, where each x_i is a pixel intensity, each logistic function in a convolutional neural network simply computes $\beta_0 + \sum_i \beta_i x_i$, where i indexes a vector of the pixel intensities of adjacent pixels, which is then linked to the output through a logistic function. By using

[4]For great introductions on the use of convolutional neural networks to classify images, we point readers to Joo and Steinert-Threlkeld (2018); Williams et al. (2020); Torres and Cantú (2022).

these window averages, we can effectively structure the neural network so that the number of parameters does not grow unnecessarily.

As a demonstration, we replicate the study in Helber et al. (2019) which classifies the land use of remotely sensed satellite images of European cities in over 34 countries. The dataset contains 27,000 images across 10 types of land use: (a) Industrial Buildings; (b) Residential Buildings; (c) Annual Crop; (d) Permanent Crop; (e) River; (f) Sea and Lake; (g) Herbaceous Vegetation; (h) Highway; (i) Pasture; (j) Forest. We present example images of land use types in Figure 4.

Figure 4: Land Use Types



(a) Industrial Building **(b)** Residential Area **(c)** Annual Crop **(d)** Permanent Crop

(e) River **(f)** Sea and Lake **(g)** Herbaceous Vegetation **(h)** Highway

(i) Pasture **(j)** Forest

We train relatively shallow convolutional neural networks (ranging from two to six hidden layers) to predict the label of land use on these images.[5] Empirically, we find that increasing the number of hidden layers improves the classification accuracy while also increasing the time it takes to train the model. The best-performing model in our application, the convolutional neural network with six hidden layers, achieves an accuracy rate of 89% in out-of-sample image classification when splitting the dataset to 70% for training and 30% for validation.

Though a stylized illustration, this application demonstrates potential avenues for future research in the social sciences. For example, scholars interested in conflict and borders might use available satellite imagery to identify changes in land suitability for farming before and after conflict (Lin, 2022a), or to study the origins of state borders (Carter and Goemans, 2011).

4.2 A Recurrent Neural Network on Hate Speech

While the previous application demonstrated how we can use the structure of neural networks to model spatial dependence in data, we can also structure neural networks to suit other tasks. For example, instead of modeling spatial dependence, recurrent neural networks model temporal dependence. This is useful in a range of prediction tasks. For example, in text classification, it is common to discard word order (the bag-of-words assumption). Recurrent neural networks, however, naturally respect the temporal ordering of terms in a document, and can recognize occurrences like negation, where the adjacency of terms is critical for their proper interpretation.

Recurrent neural networks can also be applied to any classification problem where the data are a time series. Models for time series are of course tremendously common in the social sciences. Hidden Markov models, for instance, have a long history in quantitative political science when modeling temporally dependent data (Schrodt, 1997; Park, 2012; Knox and Lucas, 2021), and time series analysis more generally is one of the most common empirical approaches in the discipline (Beck and Katz, 1995; Franzese and Hays, 2007; Lebo and Box-Steffensmeier, 2008; Slapin and Proksch, 2008; Box-Steffensmeier et al., 2014).

In fact, a hidden Markov model can be represented as a special case of a recurrent neural

[5]As noted in Section 3.2, neural networks can in fact be composed of many hidden layers. And due to their flexibility, there is tremendous variation in the structure of convolutional neural networks, and best practices continue to evolve. Though beyond the scope of this chapter, Torres and Cantú (2022) discusses convolutional neural networks in greater detail. On this data in particular, Helber et al. (2019) evaluate a range of different approaches to the construction of convolutional neural networks.

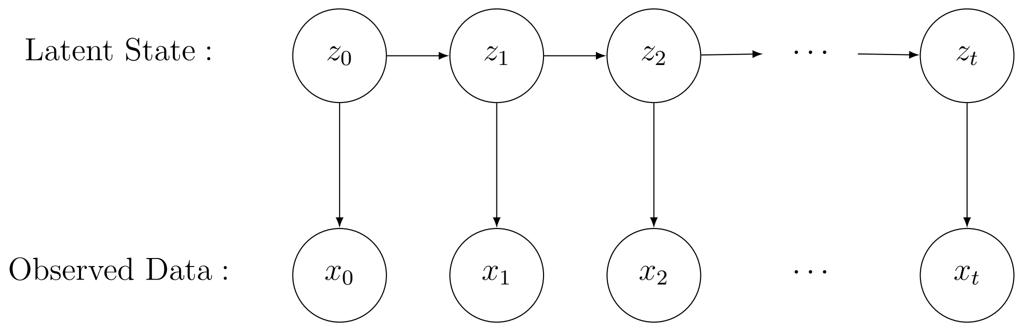


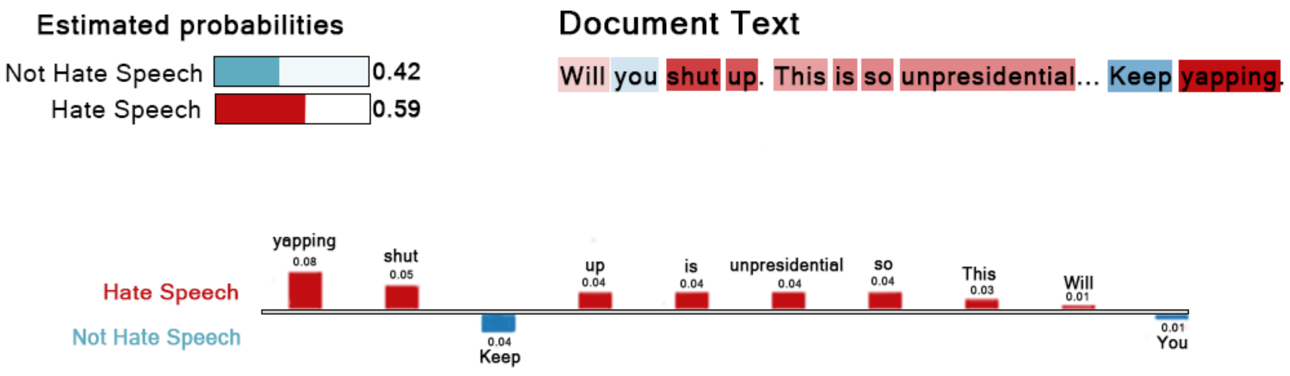
Figure 5: Data generating process of a recurrent neural network. A latent state z_t generates observable data x_t, where z_t is a function of z_{t-1}.

network (Sun et al., 1990), and the general structure is similar at an abstract level. Both suit a data generating process where a latent state z_t generates some observable data x_t, and where z_t is a function of z_{t-1}, as shown in Figure 5.

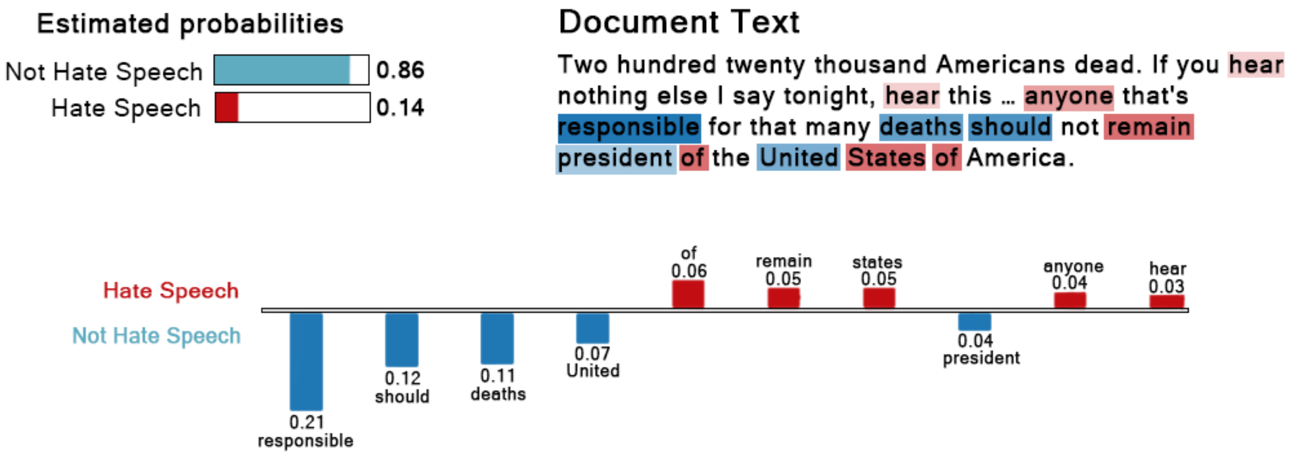
In practice, a recurrent neural network estimates the state at time t as $z_t = f(z_{t-1}, x_t)$. We demonstrate recurrent neural networks with text classification, where each document is treated as a sequence of words and the observed data x_t are the words in the document in the order in which they appear, while the latent state is a label (in this case, the latent state equals 1 if the document is an example of hate speech, otherwise 0). In other words, in this example, the latent state is constant for all time steps t in a given document, but varies across documents. However, in practice, this need not be the case. The latent state could instead identify sentiment at that particular point in text, for example, which might vary throughout a document.

In this demonstration, we use the Hateful Memes Dataset (Kiela et al., 2020), which was produced and released by Facebook AI and has been analyzed by Wu and Mebane Jr (2022), among others. The version of the Hateful Memes Dataset that we analyze contains 8,500 social media posts, each of which was labeled by human coders according to whether or not it constituted hate speech. In the subset that we study, each post has an associated text string and an image. We focus on the text data to demonstrate recurrent neural networks. Of the 8,500 posts, approximately 36% are hate speech.

This is a relatively difficult task. Kiela et al. (2020) report that trained human coders only agreed on the coding for posts 84.7% of the time. We suggest that this in part reflects uncertainty in the true coding for at least some posts, and as a result, no classifier is going



(a) Quotation A: Estimated to be hate speech.



(b) Quotation B: Not estimated to be hate speech.

Figure 6: The top and bottom figures plot two quotations from the 2020 Presidential Debate ("Document Text"). "Estimated probabilities" is the probability that the recurrent neural network assigns to the event that the quotation is hate speech or not. The barplot beneath shows the weight assigned to each term in the quotation by a simpler, more-interpretable model that summarizes how the recurrent neural network makes predictions at the local point of the specific quotation.

to achieve perfect accuracy since the labels are imperfect. We fit a simple recurrent neural network to this data, which takes as input the first word, updates the estimated state at time t, then subsequently passes that state forward and repeats again, through the entire document. The best performing model achieved 71.2% accuracy.

In Section 3.2, we noted that neural networks are famously difficult to interpret. Here, we briefly demonstrate one approach to interpreting neural networks despite their complexity. To do so, we use the method developed by Ribeiro et al. (2016), which approximates a complex model, like a recurrent neural network, with a simpler, more interpretable model *at a specific data point.* While there may be no linear model that approximates the estimated recurrent neural network, there is likely a linear model that approximates to our recurrent neural network at a specific point in the data. This is analogous to local regression (e.g., LOESS), where a complex function can be approximated at localized subsets of the data by a much simpler function.

Figure 6 visualizes the results of this exercise. We use two quotes made by President Biden in 2020 Presidential Election Debates and test whether they are classified as hate speech by the recurrent neural network that we estimate in this application. We do so for two reasons. First, while it is important to be able to identify hate speech online, there is no need to draw additional attention to hate speech. Second, interpreting the model on these quotations is arguably more interesting than interpreting it on the data on which it was trained. Though neither quotation constitutes hate speech, the model wrongly classifies the first as hate speech. Out-of-sample and out-of-domain tests — where the model is evaluated on data from a different source than that on which it was trained — provide insight into how the model is succeeding in-sample and where it might break.

The top panel of Figure 6 shows the weight assigned to each word in the quotation by a simpler, interpretable approximation of the the neural network in the local region of the data where this document exists. In this simpler approximation, "yapping" receives the largest weight, then "shut," and so on. Intuitively, it appears that this text was wrongly classified as hate speech due to the presence of negative words like "yapping" and "shut up," which likely do appear often in hate speech. In contrast, the lower panel shows a second quote, which is correctly classified as not constituting hate speech. In this case, the simpler local approximation of the neural network gives the largest weight to the term "responsible," and also gives relatively substantial weight to "united," both of which are ostensibly rare in hate speech, at least in the training data used in this example.

While this application demonstrates one way in which recurrent neural networks can be interpreted, we note that it is still difficult. For example, if we rearranged the sentence order in the quotations above, the weights given to the local approximation change due to the sequential structure of the model. Nonetheless, although neural networks are often maligned for their lack of interpretability, much can be done to make at least some sense of the predictions they generate.

5 Words of Caution

Neural networks are increasingly common across a range of domains, particularly in classification tasks with unstructured data. In this chapter, we reviewed an increasingly common use of machine learning in the social sciences: estimation of a proxy variable. We then discussed how in the familiar setting of regression with a binary dependent variable, the trade-off between familiar statistical models shifts. With a simulation, we illustrated how neural networks approximate complex functions, and demonstrated their ability to learn from more complex data in applications to text and image classification. In particular, we drew attention to similarities between machine learning and statistical methods that are common to the social sciences.

This chapter highlights several open areas of research relating to how social scientists use computational methods to learn proxy variables for abstract theoretical concepts. Most notably, at present little attention is given to the fact that proxies are estimated with uncertainty. We note that this is similar to well-studied contexts like multiple imputation and measurement error, but despite this similarity, only rarely are attempts made to account for this error in second-stage causal tests. We point readers to Fong and Tyler (2021), which characterizes this problem and proposes one possible approach to accounting for it, using the learned proxy as an instrument for its true value.

More positively, however, developments in computational methods permit the study of many new quantities of interest. Interdisciplinary work leveraging these developments has the potential to considerably improve our understanding of long-standing questions in social science by providing new measures of tough-to-observe quantities.

References

Abdar, M., F. Pourpanah, S. Hussain, D. Rezazadegan, L. Liu, M. Ghavamzadeh, P. Fieguth, X. Cao, A. Khosravi, U. R. Acharya, et al. (2021). A review of uncertainty quantification in deep learning: Techniques, applications and challenges. *Information Fusion 76*, 243–297.

Ashworth, S., C. R. Berry, and E. B. De Mesquita (2015). All else equal in theory and data (big or small). *PS: Political Science & Politics 48*(1), 89–94.

Barari, S., C. Lucas, and K. Munger (2021). Political deepfake videos misinform the public, but no more than other fake media. *OSF Preprints. doi 10.*

Beck, N. and J. N. Katz (1995). What to do (and not to do) with time-series cross-section data. *American political science review 89*(3), 634–647.

Beck, N., G. King, and L. Zeng (2000). Improving quantitative studies of international conflict: A conjecture. *American Political Science Review 94*(1), 21–35.

Blackwell, M., J. Honaker, and G. King (2017). A unified approach to measurement error and missing data: overview and applications. *Sociological Methods & Research 46*(3), 303–341.

Box-Steffensmeier, J. M., J. R. Freeman, M. P. Hitt, and J. C. Pevehouse (2014). *Time series analysis for the social sciences.* Cambridge University Press.

Cantú, F. (2019). The fingerprints of fraud: Evidence from mexico's 1988 presidential election. *American Political Science Review 113*(3), 710–726.

Carter, D. B. and H. E. Goemans (2011). The making of the territorial order: New borders and the emergence of interstate conflict. *International Organization 65*(2), 275–309.

Chang, C. and M. Masterson (2020). Using word order in political text classification with long short-term memory models. *Political Analysis 28*(3), 395–411.

Cheng, X., B. Khomtchouk, N. Matloff, and P. Mohanty (2018). Polynomial regression as an alternative to neural nets. *arXiv preprint arXiv:1806.06850*.

Fong, C. and M. Tyler (2021). Machine learning predictions as regression covariates. *Political Analysis 29*(4), 467–484.

Franzese, R. J. and J. C. Hays (2007). Spatial econometric models of cross-sectional interdependence in political science panel and time-series-cross-section data. *Political analysis 15*(2), 140–164.

Gawlikowski, J., C. R. N. Tassi, M. Ali, J. Lee, M. Humt, J. Feng, A. Kruspe, R. Triebel, P. Jung, R. Roscher, et al. (2021). A survey of uncertainty in deep neural networks. *arXiv preprint arXiv:2107.03342*.

Gomila, R. (2021). Logistic or linear? estimating causal effects of experimental treatments on binary outcomes using regression analysis. *Journal of Experimental Psychology: General 150*(4), 700.

Goodfellow, I., Y. Bengio, and A. Courville (2016). *Deep Learning.* MIT Press. `http://www.deeplearningbook.org`.

Helber, P., B. Bischke, A. Dengel, and D. Borth (2019). Eurosat: A novel dataset and deep learning benchmark for land use and land cover classification. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing 12*(7), 2217–2226.

Hornik, K., M. Stinchcombe, and H. White (1989). Multilayer feedforward networks are universal approximators. *Neural networks 2*(5), 359–366.

Joo, J. and Z. C. Steinert-Threlkeld (2018). Image as data: Automated visual content analysis for political science. *arXiv preprint arXiv:1810.01544*.

Kellstedt, P. M. and G. D. Whitten (2018). *The fundamentals of political science research.* Cambridge University Press.

Kiela, D., H. Firooz, A. Mohan, V. Goswami, A. Singh, P. Ringshia, and D. Testuggine (2020). The hateful memes challenge: Detecting hate speech in multimodal memes. *Advances in Neural Information Processing Systems 33*, 2611–2624.

King, G., M. Tomz, and J. Wittenberg (2000). Making the most of statistical analyses: Improving interpretation and presentation. *American journal of political science*, 347–361.

Knox, D. and C. Lucas (2021). A dynamic model of speech for the social sciences. *American Political Science Review 115*(2), 649–666.

Knox, D., C. Lucas, and W. K. T. Cho (2022). Testing causal theories with learned proxies. *Annual Review of Political Science 25*, 419–441.

Lebo, M. J. and J. M. Box-Steffensmeier (2008). Dynamic conditional correlations in political science. *American Journal of Political Science 52*(3), 688–704.

LeCun, Y., Y. Bengio, and G. Hinton (2015). Deep learning. *Nature 521*(7553), 436.

Lin, E. (2022a). How war changes land: soil fertility, unexploded bombs, and the underdevelopment of cambodia. *American Journal of Political Science 66*(1), 222–237.

Lin, G. (2022b). Estimating short text similarity with a customized meaning-based approach. *Working*.

Park, J. H. (2012). A unified method for dynamic and cross-sectional heterogeneity: Introducing hidden markov panel models. *American Journal of Political Science 56*(4), 1040–1054.

Poole, K. T. and H. Rosenthal (1985). A spatial model for legislative roll call analysis. *American journal of political science*, 357–384.

Rheault, L. and C. Cochrane (2020). Word embeddings for the analysis of ideological placement in parliamentary corpora. *Political Analysis 28*(1), 112–133.

Ribeiro, M. T., S. Singh, and C. Guestrin (2016). " why should i trust you?" explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pp. 1135–1144.

Roberts, M. E., B. M. Stewart, and D. Tingley (2016). Navigating the local modes of big data. *Computational social science 51.*

Rodriguez, P. L. and A. Spirling (2022). Word embeddings: What works, what doesn't, and how to tell the difference for applied research. *The Journal of Politics 84*(1), 101–115.

Schrodt, P. A. (1997). Early warning of conflict in southern lebanon using hidden markov models. In *American Political Science Association*.

Slapin, J. B. and S.-O. Proksch (2008). A scaling model for estimating time-series party positions from texts. *American Journal of Political Science 52*(3), 705–722.

Sun, G.-Z., H. Chen, Y.-C. Lee, and C. L. Giles (1990). Recurrent neural networks, hidden markov models and stochastic grammars. In *1990 IJCNN International Joint Conference on Neural Networks*, pp. 729–734. IEEE.

Torres, M. and F. Cantú (2022). Learning to see: Convolutional neural networks for the analysis of social science data. *Political Analysis 30*(1), 113–131.

Williams, N. W., A. Casas, and J. D. Wilkerson (2020). *Images as data for social science research: An introduction to convolutional neural nets for image classification*. Cambridge University Press.

Wu, P. Y. and W. R. Mebane Jr (2022). Marmot: A deep learning framework for constructing multimodal representations for vision-and-language tasks. *Computational Communication Research 4*(1).