Holographic Feature Representations of Deep Networks

Martin A. Zinkevich
Google, Inc.

Alex Davies
Google, Inc.

Dale Schuurmans[*]
University of Alberta

Abstract

It is often asserted that deep networks learn "features", traditionally expressed by the activations of intermediate nodes. We explore an alternative concept by defining features as partial derivatives of model output with respect to model parameters—extending a simple yet powerful idea from generalized linear models. The resulting features are not equivalent to node activations, and we show that they can induce a *holographic* representation of the complete model: the network's output on given data can be exactly replicated by a simple *linear* model over such features extracted from *any* ordered cut. We demonstrate useful advantages for this feature representation over standard representations based on node activations.

1 INTRODUCTION

Deep networks provide an effective foundation for machine learning, having delivered significant advances in computer vision [10, 26, 17, 19, 8], speech recognition [7], spoken dialogue systems [9], and machine translation [9, 22]. Such models are often said to learn useful "features" [4, 23], which are normally considered to be the activations of intermediate nodes. However, we consider a more fundamental concept than node activations, and show that this alternative notion of what constitutes a "feature" can provide more powerful properties.

The standard way to think about the features learned in a deep network—as node activations across a single layer [4, 25, 26, 23]—has led to some insight into what a trained network might be representing. For example, such a perspective has led to impressive visualizations that illustrate how such models might "see" classes [20] or interpret images [15]. The layer-wise embeddings in deep networks trained on text have also been visualized and shown to encode semantically meaningful dimensions [14]. Attempts have also been made to "transfer" useful activation features between tasks [23], and to understand layer-wise representations by using them in linear classifiers [1]. In fact, this latter work bears some similarity to the present investigation, but without considering equally powerful feature representations.

Meanwhile, a key drawback of deep networks remains the stability and repeatability of training. It is well known that non-convex optimization problems can have suboptimal local minima separated from global minima, which also occurs in training deep networks [5]. It is true that local minima do not prevent good outcomes from being obtained in practice [3]; in fact, whenever one is attempting to produce a single model given a fixed training set (i.e. the "offline" scenario) any weak result can simply be discarded and training repeated. *However, the situation is quite different in the typical "online" scenario encountered in industry,* where data arrives continually and models must be constantly retrained or adjusted for user-facing applications. In such cases, training instability can at minimum be a nuisance and at worst be dangerous. The online scenario presents an important challenge for deep networks and raises some of the key questions we attempt to address in this paper. Can features be extracted from a deep network that can be reliably used in other models? Can extracted features be used to more efficiently and stably incorporate new data?

In this paper, we provide a different perspective on what a "feature" might represent about a deep network. We show how the specific features we extract can be used to recover a linear model that *exactly* reproduces the original network on the training data. These features incorporate both the "upstream" information from previous layers, and the "downstream" information from later layers, giving a complete "holographic" representation

[*]Work supported by Google.

of the network from the perspective of any given layer. An important advantage is that this representation can be rapidly re-trained to global optimality on new data, efficiently obtaining repeatable outcomes. Moreover, the features are "calibrated" in a manner that allows weaknesses in the model versus the chosen training objective to be disambiguated, as we discuss in the next section.

2 FEATURES AND CALIBRATION

To investigate whether the rich notion of "feature" from generalized linear models can be meaningfully applied to deep networks, we leverage the concept of *calibration* (which is a particular form of *moment matching*).

Consider linear least squares regression. If a feature is 1 in each example (e.g., the bias), the average label must equal the average prediction in any optimally trained model. More generally, for any $\{0,1\}$-valued (boolean) feature, the average label must equal the average prediction when the feature is 1 [16, Equation 3]. This calibration property provides a powerful diagnostic: Imagine one is trying to predict the number of minutes someone who searches for "cat videos" will watch `cat_12`. Having a feature that is 1 when the search is "cat videos" and the video shown is `cat_12` guarantees that the average prediction will equal the average label in this subset. Thus, if you wanted to rank videos by how many minutes will be watched, you can disambiguate between mistakes in the label (e.g. `cat_12` is watched more on average than `cat_13` but is a worse video by a different metric: do we need to change our metric?) and mistakes in the modeling (e.g. `cat_12` is watched more on average than `cat_13`, but predicted on average to be watched less).

A similar property holds for logistic regression after distinguishing the *predicted log odds* (a linear combination of the features) from the *predicted probability* (a function of the log odds). In an optimally trained model, the average predicted probability will equal the average label whenever a boolean feature is 1. That is, if we tried to predict the probability someone would click on video `cat_12` given that they search for "cat videos", the average predicted probability would be the average label.

This notion of calibration provides a practical tool for disambiguating problems with the feature representation versus problems with the objective. We use it as a key defining concept in the technical development below.

3 FORMALISM

Our basic strategy is to relate deep networks to generalized linear models—such as linear, logistic and Poisson regression—by considering supervised learning problems defined in terms of exponential families [16, 21].

3.1 EXPONENTIAL FAMILIES

We base our development on full, minimal, 1-dimensional, standard exponential families [2] (see Appendix A for standard definitions), which will be used to formulate the training loss. We will denote the exponential family being used by p, from which one can derive the domain $\Omega_p \subseteq \mathbf{R}$ of possible outcomes, the mean function $\mu_p : \mathbf{R} \to \mathbf{R}$ of the outcome given the parameter, and the loss $\ell_p : \Omega_p \times \mathbf{R} \to \mathbf{R}$ given by the negative log likelihood of the outcome given the parameter.

For example, the Bernoulli family is specified by $\Omega_p = \{0,1\}$, $\ell_p(\omega,\theta) = -\theta\omega + \ln(1+\exp(\theta))$, and $\mu_p(\theta) = \frac{1}{1+\exp(\theta)}$; this family forms the basis of logistic regression. Other examples, such as Gaussian with a fixed variance (least squares regression), and the Poisson family (Poisson regression) are given in Appendix A. The following lemma encapsulates the key facts we require:

Lemma 1 *The Bernoulli family, Gaussian family (fixed variance), and Poisson family are full, minimal, 1-dimensional standard exponential families. For a full, minimal, 1-dimensional standard exponential family p:*

1. *for every $\theta \in \mathbf{R}$, $\omega \in \Omega_p$, $\ell_p(\omega,\theta)$ is differentiable with respect to θ and $\frac{\partial \ell_p(\omega,\theta)}{\partial\theta} = \mu_p(\theta) - \omega$; and*
2. *ℓ_p is **strictly convex in its second argument**: for every $\theta,\theta' \in \mathbf{R}$, if $\theta \neq \theta'$, then for all $\lambda \in (0,1)$ we have $\ell_p(\omega, \lambda\theta + (1-\lambda)\theta') < \lambda\ell_p(\omega,\theta) + (1-\lambda)\ell_p(\omega,\theta')$.*

Proofs for all lemmas and theorems stated in this paper are given in the appendices.

3.2 SUPERVISED LEARNING PROBLEM

Our focus in this paper will be on supervised learning problems. A **model** is defined to be a function $M : X \to \mathbf{R}$. A **model family** is given by a parameterized set of models $\mathcal{M} = \{M_w\}_{w\in\mathbf{R}^S}$ where S is a finite set of parameters. We define a **supervised learning problem** (or just a **problem** for short) to be a tuple $\mathcal{P} = (p, X, \{x_1 \dots x_m\}, \{y_1 \dots y_m\})$, where p is a full, minimal, 1-dimensional standard exponential family, X is the domain of instances, $x_1 \dots x_m \in X$ are the instances, and $y_1 \dots y_m \in \Omega_p$ are the labels. We define a **loss function** $L_{\mathcal{P}} : \mathbf{R}^X \to \mathbf{R}$ that maps a model M to a scalar loss value via:

$$L_{\mathcal{P}}(M) = \sum_{i=1}^{m} \ell_p(y_i, M(x_i)), \tag{1}$$

the negative log likelihood of the labels given instances.

3.3 CALIBRATION, FEATURES, OPTIMALITY

Definition 2 *Given a supervised learning problem $\mathcal{P} = (p, X, \{x_1 \dots x_m\}, \{y_1 \dots y_m\})$,*[1] *a **feature** is a function $f : X \to \mathbf{R}$. We say that a model M is **calibrated with respect to feature f on $\mathcal{P}$** if:*

$$\sum_{i=1}^{m} f(x_i)\mu_p(M(x_i)) = \sum_{i=1}^{m} f(x_i)y_i. \tag{2}$$

For example, if the feature $f : X \to \mathbf{R}$ is 1 if the instance comes from the United States, and 0 otherwise, then calibration of a model with respect to this feature would imply that the average prediction given an instance from the United States would equal the average label for an instance from the United States. Note that the concept of calibration with respect to a feature is independent of whether the feature is actually used by the model M. In the literature on regression, there are many results that show how various generalized linear regression problems find models that are calibrated with respect to the features [16, 21]. In this paper, we will consider starting with a model and generating calibrated features from it.

Definition 3 *For a model family $\{M_w\}_{w \in \mathbf{R}^S}$ with $w \in \mathbf{R}^S$ and $s \in S$, we define the **feature generated from parameter s of model M_w** to be $f_s : X \to \mathbf{R}$ such that:*

$$f_s(x) = \frac{\partial^+ M_w(x)}{\partial w_s} \quad \textit{for all } x \in X; \tag{3}$$

in other words, a feature is specified by the right partial derivative of the output of model M_w acting on x with respect to some parameter s.[2]

Definition 4 *Given a parameter subset $S' \subseteq S$, the features generated from S' of M_w are **total for $x \in X$** if:*

$$\lim_{h \to 0} \frac{\left|M_{w+h}(x) - \left(M_w(x) + \sum_{s' \in S} h_{s'} f_{s'}(x)\right)\right|}{\|h\|} = 0 \tag{4}$$

*(note $h \in \mathbf{R}^{S'}$), where $f_{s'}$ is the feature generated from s' of model M_w, and $w + h \in \mathbf{R}^S$ obeys $(w+h)_{s'} = w_{s'} + h_{s'}$ if $s' \in S'$, and $(w+h)_{s'} = w_{s'}$ if $s' \notin S'$. For brevity, we will also call such a set S' itself **total**.*[3]

First note that the property of a feature set S' being total exhibits downward inheritance: if S' is total for some x, then so is S'' for any $S'' \subseteq S'$. Therefore, if $M_w(x)$ is differentiable at w, implying that the full feature set S is total, then any generated feature set must also be total. Note that it is not sufficient that $M_w(x)$ be be partially differentiable with respect to w to ensure totality; specifically, if the features generated from S' are total, and the features generated from S'' are total, there is no guarantee that the features generated from $S' \cup S''$ are total—Appendix E provides a concrete example.

Definition 5 *Given a problem $\mathcal{P} = (p, X, \{x_1 \dots x_m\}, \{y_1 \dots y_m\})$, two models M and M' are **equal on the training data** if for all $i \in \{1 \dots m\}$, $M(x_i) = M'(x_i)$.*

Definition 6 *Given a problem $\mathcal{P}$ and a model family $\mathcal{M}$, we say that $M \in \mathcal{M}$ is **optimal** for $\mathcal{P}$ on $\mathcal{M}$ if, for all $M' \in \mathcal{M}$, $L_{\mathcal{P}}(M) \le L_{\mathcal{P}}(M')$.*

Notice that optimality is based upon both the problem and the model family: this will become important later when we talk about models outside of a family being equivalent to optimal models inside a family.

In what follows we will distinguish these generated features from the more common node activations. In some cases they are the same, as in the last layer, other times they are related, and sometimes they are completely different, as in the generated features from bias parameters.

4 GENERATED FEATURES

We first show that if the partial derivative of the loss with respect to a parameter is zero, then the feature generated by that parameter must be calibrated.

Theorem 7 *For a problem $\mathcal{P}$ and model family $\mathcal{M} = \{M_w\}_{w \in \mathbf{R}^S}$, given a $w \in \mathbf{R}^S$ and $s \in S$ such that f_s is the feature generated from parameter s of model M_w: if $\{s\}$ is total on the training data given M_w and $\frac{\partial L_{\mathcal{P}}(M_w)}{\partial w_s} = 0$, then M_w is calibrated with respect to f_s.*

Note that we do not assume that the model family $\mathcal{M}$ is *linearly* parameterized; far from it, below we will be considering model families defined by certain classes of deep neural networks. Nevertheless, the features generated in this way can be exported to an external family of linear models defined over the same feature set. Such an auxiliary family of linear models will possess several strong properties with respect to the original nonlinear model family, as we now show.

Definition 8 *For a finite S, given a set of functions $\{f_s\}_{s \in S}$ where $f_s : X \to \mathbf{R}$, we define the **family of linear models induced by $\{f_s\}_{s \in S}$**, to be the model family*

[1] When we mention a feature in the context of a model or problem, we assume the feature shares the same domain.

[2] Note that this definition is related but not identical to the Fisher score, given by $\partial \log \ell_p(y, M_w(x))/\partial w_s$. A key difference is that the Fisher score requires the label y, whereas a generated feature is only defined with respect to model output.

[3] Appx. I.1 shows how totality relates to differentiability.

$\{N_w\}_{w\in\mathbf{R}^S}$ *such that for all* $w \in \mathbf{R}^S$ *and* $x \in X$*:*

$$N_w(x) = \sum_{s\in S} w_s f_s(x). \tag{5}$$

Let this family be denoted by $\mathcal{L}(\{f_s\}_{s\in S})$.

Note that if we were generating features from a model M_w that was already linearly parameterized, we would simply re-generate the original feature set. Next, we need some preliminary results for families of linear models.

Lemma 9 *For finite* S*, given a set of features* $\{f_s\}_{s\in S}$*, the family of linear models* $\mathcal{L}(\{f_s\}_{s\in S})$*, and* $N_w \in \mathcal{L}$*: the set* S *is total for all* $x \in X$ *given* N_w*, and the feature generated from parameter* s *by model* N_w *is* f_s.

Lemma 10 *(c.f. [16, Equation 10]) Given a problem* $\mathcal{P}$*, a finite set* S *and a set of features* $\{f_s\}_{s\in S}$*: a model* N *in the family of linear models* $\mathcal{L}(\{f_s\}_{s\in S})$ *is optimal if and only if* N *is calibrated with respect to* f_s *for all* $s \in S$.

Lemma 11 *Given a problem* $\mathcal{P}$*, a finite set* S*, a subset* $S' \subseteq S$*, and a set of features* $\{f_s\}_{s\in S}$*: if a model* N *is optimal in the family of linear models* $\mathcal{L}(\{f_s\}_{s\in S'})$ *and* N *is calibrated with respect to* f_s *for all* $s \in S - S'$*, then* N *is also an optimal model in* $\mathcal{L}(\{f_s\}_{s\in S})$.

Now consider a general model family $\mathcal{M}$. Even if the models in $\mathcal{M}$ are not linear, our first key result is that a family of linear models defined over the features *generated* from $\mathcal{M}$ (via Definition 3) will satisfy important properties.

Definition 12 *Given a finite* S*, family* $\{M_w\}_{w\in\mathbf{R}^S}$ *of models, subset* $S' \subseteq S$*, and parameters* $w \in \mathbf{R}^{S'}$*: if* f_s *is the feature generated by* s *given* M_w*, then the* ***family of linear models associated with*** $\boldsymbol{S'}$ ***given*** $\boldsymbol{M_w}$ *will be denoted* $\mathcal{L}(\{f_s\}_{s\in S'})$.

Lemma 13 *Given a problem* $\mathcal{P}$*, a finite set* S*, a subset* $S' \subseteq S$*, a model family* $\{M_w\}_{w\in\mathbf{R}^S}$*, and a* $w \in \mathbf{R}^S$ *such that* $\frac{\partial L_{\mathcal{P}}(M_w)}{\partial w_s} = 0$ *for all* $s \in S'$*:* ***if*** $\mathcal{L}'$ *is the family of linear models associated with* S' *given* M_w*,* S' *is total on the training data given* M_w*, and* M_w *is equal on the training data to some* $N' \in \mathcal{L}'$*,* ***then*** N' *is optimal in* $\mathcal{L}'$.

This next lemma goes one step further, by showing that if a model $M \in \mathcal{M}$ is stationary on a subset of parameters S' and if there is a linear model over features generated on S' that matches M, then there is an optimal linear model over features generated on S that also matches M.

Lemma 14 *Given a problem* $\mathcal{P}$*, a finite set* S*, a subset* $S' \subseteq S$*, a model family* $\{M_w\}_{w\in\mathbf{R}^S}$*, and a* $w \in \mathbf{R}^S$ *such that* $\frac{\partial L_{\mathcal{P}}(M_w)}{\partial w_s} = 0$ *for all* $s \in S$*:* ***if*** $\mathcal{L}'$ *is the family of linear models associated with* S' *given* M_w*,* S' *is total,* M_w *is equal on the training data to some* $N' \in \mathcal{L}'$*, and* $\mathcal{L}''$ *is the family of linear models associated with* S*,* ***then*** *any optimal* $N'' \in \mathcal{L}''$ *equals* M_w *on the training data.*

So a key question will be to understand when a model M_w that is stationary on a subset S' of parameters will also be equal on the training data to a stationary model that uses all the parameters S. To address deep neural networks we will introduce the concept of *homogeneity*, which will allow us to prove that, for certain families of deep models, if a model is stationary with respect to a particular subset of its parameters it will also be equal on the training data to a linear model over the same subset. We will use this construction to show how the generated features can capture *all* of the expressiveness of a given deep model with respect to the training data, while still allowing one to re-train in a linear setting.

5 FEEDFORWARD NETWORKS

We first need to formally define deep neural networks for subsequent analysis. For generality we follow the formalization given in [18] that extends the more conventional layered representation; see also Appendix B.

A **feedforward neural network** D is defined by a directed acyclic graph with objects attached to the vertices and edges: in particular, $D = (V, E, I, \{g_i\}_{i\in I}, o^*, A)$, where V is a set of vertices, $E \subseteq V \times V$ is a set of edges, $I = \{i_1 \dots i_m\} \subset V$ is a set of input vertices, $g_i : X \to \mathbf{R}$ is an input function connected to $i \in I$, $o^* \in V$ is the output vertex, and $A = \{a_v : v \in V\}$ is a set of activation functions, $a_v : \mathbf{R} \to \mathbf{R}$. The parameters are $w \in \mathbf{R}^E$.

We will need to refer to the partial ordering on vertices implied by $G = (V, E)$, where we assume G contains no cycles,[4] the input vertices have no incoming edges (i.e. $(u, i) \notin E$ for all $i \in I$, $u \in V$), the output vertex is not an input (i.e. $o^* \notin I$) and the output vertex has no outgoing edges (i.e. $(o^*, v) \notin E$ for all $v \in V$). A directed acyclic graph defines a partial order $\leq$ on vertices where $u \leq v$ if and only if there is a path from u to v.

Given a training input $x \in X$, the computation of the network D is specified by a circuit function $c_{v,w}$ that assigns values to each vertex based on the partial order:

$$c_{i,w}(x) = a_i(g_i(x)) \text{ for } i \in I; \tag{6}$$

$$c_{v,w}(x) = a_v\Big(\sum_{u:(u,v)\in E} c_{u,w}(x) w_{(u,v)}\Big) \text{ for } v \in V - I. \tag{7}$$

[4] A **path** $(v_1, \dots, v_k)$ is a sequence of vertices such that $(v_j, v_{j+1}) \in E$ for all j. A **cycle** is a path with $v_1 = v_k$.

Note that the activations on input and output nodes are usually the identity, i.e. $a_v(x) = x$ for $v \in I \cup \{o^*\}$. We can also add a bias vertex $b \in I$, with input function $g_b(x) = 1$ for all $x \in X$, so that adding an edge $(b, v) \in E$ ensures that vertex v receives an affine rather than a linear combination of its incoming circuit values.

Definition 15 *Given $D = (V, E, I, \{g_i\}_{i \in I}, o^*, A)$, we will denote the* ***family of feedforward network models*** *by $\mathcal{D}(V, E, I, \{g_i\}_{i \in I}, o^*, A) = \{M_w\}_{w \in \mathbf{R}^E}$, where for all $w \in \mathbf{R}^E$ we let $M_w = c_{o^*,w}$.*

6 HOMOGENEOUS MODEL FAMILIES

Our main theoretical development applies to feedforward neural networks with *homogeneous* activation functions.

Definition 16 *A function $f : \mathbf{R}^p \to \mathbf{R}^q$ is* ***homogeneous*** *(of degree 1) [11] if for all $v \in \mathbf{R}^p$ and all $\lambda \geq 0$:*

$$f(\lambda v) = \lambda f(v). \tag{8}$$

Homogeneity implies that a function is linear along any ray from the origin, and also that it can be decomposed as an inner product between inputs and partial derivatives.

Lemma 17 (Euler's Homogeneous Function Theorem) (Degree 1 Case) *If a homogeneous function $f : \mathbf{R}^p \to \mathbf{R}^q$ is differentiable at $x \in \mathbf{R}^p$, then for all $k \in \{1 \ldots q\}$,*

$$f_k(x) = \sum_{j=1}^{p} \frac{\partial f_k(x)}{\partial x_j} x_j. \tag{9}$$

An important example of a (degree 1) homogeneous function is the standard neural network activation function $relu : \mathbf{R} \to \mathbf{R}$, given by $relu(x) = \max(x, 0)$.

Fact 18 *The $relu$ and leaky $relu$ [13] activation functions are homogeneous. Any linear function is homogeneous, therefore so are projection and identity. The sum or composition of homogeneous functions is homogeneous, therefore constant multiplication of a homogeneous function is also homogeneous.*

We can now state our main results, first in terms of general families of homogeneous functions.

Definition 19 *Given a finite set S, a subset $S' \subseteq S$, and an instance space X: we say that a model family $\{M_w\}_{w \in \mathbf{R}^S}$ is* ***homogeneous on the parameter set S'*** *if for all $w \in \mathbf{R}^S$, $x \in X$ and $\lambda \geq 0$,*

$$M_v(x) = \lambda M_w(x), \tag{10}$$

where $v \in \mathbf{R}^S$, $v_s = \lambda w_s$ when $s \in S'$, and $v_s = w_s$ when $s \notin S'$.

Note that under this definition homogeneity is a property of a set of parameters not an individual parameter: if a model family is homogeneous on S' and on S'', that does not imply it is homogeneous on $S' \cup S''$; however, if $S''' \subseteq S'$, it will be homogeneous on S'''.

Our main results show that homogeneity and totality allow a holographic feature set to be extracted, where an auxiliary linear model on the feature set can replicate the output of any model in the family over the training data.

Theorem 20 *If a model family $\mathcal{M}$ is homogeneous on the parameter set S', $M \in \mathcal{M}$, $\mathcal{L}'$ is the family of linear models associated with S' given M, and S' is total on the training data given M, then there exists $N \in \mathcal{L}'$ such that M and N are equivalent on the training data.*

Theorem 21 *Given a problem $\mathcal{P}$, if a model family $\{M_w\}_{w \in \mathbf{R}^S}$ is homogeneous on the parameter set S', then for any $w \in \mathbf{R}^S$:*

1. *if $\mathcal{L}'$ is the family of linear models associated with S' given M_w, S' is total on the training data given M_w, and $\frac{\partial L_{\mathcal{P}}(M_w)}{\partial w_s} = 0$ for all $s \in S'$, then M_w is equal on the training data to any optimal N' in $\mathcal{L}'$.*
2. *if $\mathcal{L}'$ is the family of linear models associated with S given M_w, S' is total on the training data given M_w, and $\frac{\partial L_{\mathcal{P}}(M_w)}{\partial w_s} = 0$ for all $s \in S$, then M_w is equal on the training data to any optimal N' in $\mathcal{L}'$.*

Theorem 21 can be proved simply by combining Theorem 20 with Lemmas 13 and 14. We thus reach the conclusion that homogeneity allows one to generate features and build an auxiliary linear model that can behave equivalently to the original model over the training data.

6.1 APPLICATION TO NEURAL NETWORKS

To apply these results to deep neural networks we need to consider the question of when a model family defined by a network specification can be homogeneous in the sense of Definition 19. Recall the definition from Section 5 where a feedforward neural network is defined in terms of a directed acyclic graph $G = (V, E)$. A **cut** of G is a partition of the vertices in V into two disjoint subsets, where a **cut set** is the set of edges in E between those two sets. We wish to partition G into two sets, $B \subseteq V$ (bottom) and $T \subseteq V$ (top), such that $B \cap T = \emptyset$, $I \subseteq B$ and $o^* \in T$. The cut set consists of edges (u, v) where $u \in B$ and $t \in T$. We will call such a cut an **ordered cut**, since $t \nleq b$ for all $b \in B$, $t \in T$ under the partial order generated by G. Note that, since there are no incoming edges to I and $o^* \notin I$ is assumed, we have the following.

Lemma 22 *For any feedforward neural network, there is an ordered cut, specifically $B = I$ and $T = V \backslash I$.*

Features generated by the parameters on edges in the cut set turn out to be of critical importance. In particular, by considering a partition of the vertices in a feedforward network into an ordered cut—consisting of B, T and the cut set—we reach the deepest result in this paper: that the solutions to the deep network can be mimicked by the solutions to the corresponding linear model defined on the features generated by the ordered cut.

Theorem 23 *Given a problem $\mathcal{P}$ and a family of feedforward network models $\mathcal{D}(V, E, I, \{g_i\}_{i \in I}, o^*, A) = \{M_w\}_{w \in \mathbf{R}^E}$, where all $a \in A$ are homogeneous: **if** B and T is an ordered cut of the feedforward network, such that E' is the cut set, **then**:*

1. *$\{M_w\}_{w \in \mathbf{R}^E}$ is homogeneous on E';*
2. *for some $w \in \mathbf{R}^E$, if E' is total on the training data given M_w, $\mathcal{L}'$ is the family of linear models associated with E' given M_w, and $\frac{\partial L_{\mathcal{P}}(M_w)}{\partial w_e} = 0$ for all $e \in E'$, then M_w is equal on the training data to any optimal model in $\mathcal{L}'$;*
3. *for some $w \in \mathbf{R}^E$, if $\mathcal{L}'$ is the family of linear models associated with E given M_w, if E is total on the training data given M_w, and $\frac{\partial L_{\mathcal{P}}(M_w)}{\partial w_e} = 0$ for all $e \in E$, then M_w is equal on the training data to any optimal model in $\mathcal{L}'$.*

This result is of particular significance to the problem of repeated training, where data is arriving over time and one needs to continually produce new models. In such a scenario, it is important that the models produced behave stably: roughly speaking, there needs to be some form of repeatability in the training process and consistency in the resulting model prediction errors. Such stability is important so that human beings, who must ultimately decide whether a machine learning system can be trusted in a continuous setting, can accept that a subsequent trained model will not make significantly worse mistakes than a trained model makes today. An auxiliary family of linear models improves this notion of stability in two key ways:

1. One can approximate an optimal model with techniques from convex optimization (when it exists).
2. All optimal models are semantically equivalent on the training data.

Thus, we obtain a new form of reproducibility in a continual learning process. Such stability does not imply that the model will remain unchanged in the future, and the data may change, but this provides a solid foundation. Although the features associated with all parameters can be extracted, the fact that one can extract an equivalent linear model over a fraction of the features is a significant result. If one extracted all features, multiple distinct linear models would be optimal on the training data yet differ on new data, which would decrease stability.

It is pointed out in [21], Theorem 3.4, that there is a duality between entropy maximization and maximum likelihood. In fact, if we wish to maximize entropy subject to a set of constraints, then the member of an exponential family that satisfies those constraints maximizes entropy. Thus, when a linear model is calibrated on all of its input features, it is not only maximizing likelihood, it is maximizing entropy.[5] If a deep network is at a local minimum where the derivative is equal to zero, it is equal to one of these linear models: thus, one can consider a locally optimal model in a deep network to be maximizing entropy subject to being calibrated on the extracted features.

7 OTHER ARCHITECTURES

Not all deep models are feedforward networks: recursive neural networks are not defined on a fixed graph, residual networks (ResNets) [8] fix some weights to be constant, and convolutional neural networks (CNNs) [12] have sets of edges with equal weights.

We can handle convolutional and residual neural networks by considering a function $w^* : \mathbf{R}^n \to \mathbf{R}^E$, where we choose $v \in \mathbf{R}^n$ and use $w = w^*(v)$ as the parameters for the network. Thus, for $\mathcal{D}(V, E, I, \{g_i\}_{i \in I}, o^*, A)$ represented as $\{M_w\}_{w \in \mathbf{R}^E}$, we can write $\{Q_v\}_{v \in \mathbf{R}^n}$ such that $Q_v = M_{w^*(v)}$ for all $v \in \mathbf{R}^n$. For ResNets, we need to specify some $E^f \subseteq E$ and $w^f \in \mathbf{R}^{E^f}$, where w^f_e is always the weight of $(w^*(v))_e$. Define $E^d = E - E^f$. For CNNs, we need to specify a partition of the edges: it is easiest to do so with a function $\pi : E^d \to \{1 \ldots n\}$. So, for any $v \in \mathbf{R}^n$ and all $e \in E$:

$$(w^*(v))_e = \begin{cases} w^f_e & \text{if } e \in E^f \\ v_{\pi(e)} & \text{otherwise} \end{cases} \tag{11}$$

Thus, we will consider a **family of RC neural networks**[6] $\mathcal{RC}(V, E, I, \{g_i\}_{i \in I}, o^*, A, w^*) = \{Q_v\}_{v \in \mathbf{R}^n}$. Notice that now, instead of having $|E|$ we have n parameters. Define $E_1 \ldots E_n \subseteq E^d$ such that $E_s = \pi^{-1}(s)$, i.e. the conventional representation of the partition. Given a

[5] There is a nuance: this particular kind of entropy maximization is with respect to a particular measure. For the Bernoulli or any multinomial family, this corresponds with the traditional concept of entropy (entropy with respect to the counting measure). However, conventional differential entropy, which is defined with respect to the Lebesgue measure, is maximized subject to an additional constraint on variance.

[6] RC stands for ResNet and CNN.

family of RC feedforward neural networks, a cut B, T is **well-behaved** if it is an ordered cut and there exists an $S \subseteq \{1 \dots n\}$ such that the cut set equals $\bigcup_{s \in S} E_s$.

Theorem 24 *For a problem $\mathcal{P}$ and family of RC feedforward network models $\mathcal{RC}(V, E, I, \{g_i\}_{i \in I}, o^*, A, w^*)$ denoted $\{Q_v\}_{v \in \mathbf{R}^n}$, where all $a \in A$ are homogeneous:* ***if*** *$E_1 \dots E_n$ is the partition of the dynamic parameters, and B and T are a well-behaved cut such that there is an $S \subseteq \{1 \dots n\}$ where $E' = \bigcup_{s \in S} E_s$ is the cut set;* ***then***

1. *$\{Q_v\}_{v \in \mathbf{R}^n}$ is a homogeneous model family with respect to S;*
2. *if for some $v \in \mathbf{R}^n$, $\frac{\partial L_{\mathcal{P}}(Q_v)}{\partial v_s} = 0$ for all $s \in S$, the set S is total on the training data given Q_v, and $\mathcal{L}'$ is the family of linear models associated with S given Q_v, then Q_v is equal on the training data to any optimal N' in $\mathcal{L}'$;*
3. *if for some $v \in \mathbf{R}^n$, $\frac{\partial L_{\mathcal{P}}(Q_v)}{\partial v_s} = 0$ for all $s \in \{1 \dots n\}$, the set $\{1 \dots n\}$ is total on the training data given Q_v, and $\mathcal{L}'$ is the family of linear models associated with $\{1 \dots n\}$ given Q_v, then Q_v is equal on the training data to any optimal N' in $\mathcal{L}'$.*

Thus, if the cut exists, we can extract all the features.

8 REGULARIZATION

To this point, we have assumed parameters can have any value, and there is no external cost to choosing one set of parameters over another. However, there is evidence that regularization yields better generalization [6, Chapter 7].

In our framework, regularization can be defined in a generic way. Given a model family $\mathcal{M}$, a regularization function $R : \mathcal{M} \to \mathbf{R}$ assigns a cost to each model, independent of how it fits the data.

Definition 25 *Given a model family $\mathcal{M}$, we say that $M \in \mathcal{M}$* ***is optimal in $\mathcal{M}$ given the problem $\mathcal{P}$ and a regularization function $R : \mathcal{M} \to \mathbf{R}$****, if for all $M' \in \mathcal{M}$, $L_{\mathcal{P}}(M) + R(M) \leq L_{\mathcal{P}}(M') + R(M')$.*

This kind of optimality can sometimes be interpreted as maximum a posteriori rather than maximum likelihood optimization. For the remainder of this section, we assume we have a finite S and model family $\mathcal{M} = \{M_w\}_{w \in \mathbf{R}^S}$. For a regularization function $R : \mathcal{M} \to \mathbf{R}$, define $R^* : \mathbf{R}^S \to \mathbf{R}$ such that $R^*(w) = R(M_w)$. We assume R is (strictly) convex if R^* is (strictly) convex. Given a subset $S' \subseteq S$, we will say that a function $r : \mathbf{R}^S \to \mathbf{R}$ is **additively separable** with respect to S' if there exists $r^{S'} : \mathbf{R}^{S'} \to \mathbf{R}^S$, and $r^{S-S'} : \mathbf{R}^{S-S'} \to \mathbf{R}^S$ such that for all $w \in \mathbf{R}^S$, $r(w) = r^{S'}(\pi^{S \to S'}(w)) + r^{S-S'}(\pi^{S \to S-S'}(w))$. R is additively separable with respect to S' if R^* is additively separable with respect to S'. We can consider when, for all $s \in S$, the partial derivative with respect to s is zero:

$$\frac{\partial L_{\mathcal{P}}(M_w) + R(M_w)}{\partial w_s} = 0. \tag{12}$$

This is a stationary point and might be a saddle point or a local minima. The most common regularizers are L1 (where $R(M_w) = \sum_{s \in S} |w_s|$) and L2 (where $R(M_w) = \sum_{s \in S} w_s^2$), but group lasso [24] and other regularizations are also possible. Note that L1 regularization and group lasso are convex, L2 regularization is strictly convex, L1 and L2 regularization are separable, and group L1 regularization is sometimes separable.

As before, we want to connect homogeneous families (such as some deep networks) with regularization to the associated linear family with regularization.

Theorem 26 *Given a problem $\mathcal{P}$, a set S, a subset $S' \subseteq S$, a model family $\mathcal{M} = \{M_w\}_{w \in \mathbf{R}^S}$ that is homogeneous with respect to S', a strictly convex regularization function $R : \mathcal{M} \to \mathbf{R}$ that is additively separable with respect to S', and a model M_w where for all $s \in S'$:*

$$\frac{\partial [L_{\mathcal{P}}(M_w) + R(M_w)]}{\partial w_s} = 0, \tag{13}$$

if *S' is total on the training data given M_w, $\mathcal{L}' = \{N_v\}_{v \in \mathbf{R}^{S'}}$ is the family of linear models associated with S' given M_w, and a new regularizer $R' : \mathcal{L}' \to \mathbf{R}$ is defined such that $R'(N_v) = (R^*)^{S'}(v)$,* ***then*** *M_w must be equal on the training data to any optimal $N \in \mathcal{L}'$ given the supervised learning problem and regularization R'.*

Thus, in all circumstances where we showed functions were homogeneous (for the generic feedforward networks and some CNNs and ResNets), we now know that for a separable regularizer we can map these to a linear model family. However, we cannot simply add back all the other generated features not present in the ordered cut, because there might be a parameterization across all the generated features that has the same predictions and lower regularization. This requires further study.

9 EXPERIMENTAL EVALUATION

We investigated the main assertions that:

1. In practice, the holographic features generated can faithfully recover the original classifier.
2. The holographic features generated are indeed calibrated with respect to the original classifier.
3. Training with the generated holographic features is more stable than training a neural network.

Data set	# examples	input dim	λ	layer width	hidden layers
Pima	750 of 768	9	1/720	4	2
Census	30000 of 32561	108	1/1490	10	3
MNIST-3v5	9000 of 11552	784	1/1490	100	3

Table 1: Data sets and neural network architectures used.

4. The holographic features support generalization beyond the training set they were generated from.

We used three binary classification data sets, UCI Pima, UCI Census and MNIST (3 vs 5). The details of each data set and the respective neural network architectures used are given in Table 1. Since our experimental design required us to partition each data set into three disjoint subsets, we extracted the number of examples indicated from the initial portion of the original data then split the chosen data into three equal sized subsets. In particular, we consider training the neural network on the first third, training the extracted linear models on the middle third, and using the final third to assess generalization performance. We also normalized the input features in each case by subtracting the means and dividing by the range.

Faithfulness First we evaluate whether a linear classifier trained on holographic features can faithfully reproduce the predictions of the original neural network, whereas using activation features fails to do so. Specifically, we trained the neural network on the first data partition, extracted the holographic and activation feature sets for each layer-wise cut of the neural network, trained a linear model over these feature sets on the same data, then compared the predictions on the same training data against those made by the source neural network. Figures 1 to 3 show the agreement plots for the Pima, Census and MNIST data sets respectively. The plots show the results achieved by trained linear models over the activation and holographic feature sets respectively, as well as using exact weights for the holographic features. There are minor differences due to the neural network not being at a true local minimum, but the assertion appears to be verified.

Calibration Next, we investigate the assertion that the holographic features must be calibrated. To do so, we plot $\sum_{i=1}^{m} f(x_i)\mu_p(M(x_i))$ against $\sum_{i=1}^{m} f(x_i)y_i$ for each extracted set of holographic features over the training data. Figure 4 shows that the holographic features are indeed well calibrated, regardless of which cut was used to generate them, or which data set is considered.

Stability To determine whether a linearized representation improves learning stability compared to re-training a neural network, we conducted the following experiment. We trained a neural network on the first partition, extracted distinct holographic and activation feature sets for each layer-wise cut, then repeatedly re-trained the linear models with different random initializations on the second data partition, gathering the predictions made on the third data partition. We then re-trained the neural network on the original training data with different random initializations, and gathered the predictions made on the third data partition. To evaluate learning stability, we measured the average symmetrized KL divergence between the predictions of the different re-trained models for each type. The results are reported in Table 2. As expected, re-training produces nearly identical predictions for the linear models since the problem is convex. However, neural network re-training results in non-negligible variability between the different learned models.

Generalization Finally, we investigate whether holographic features support generalization to data from

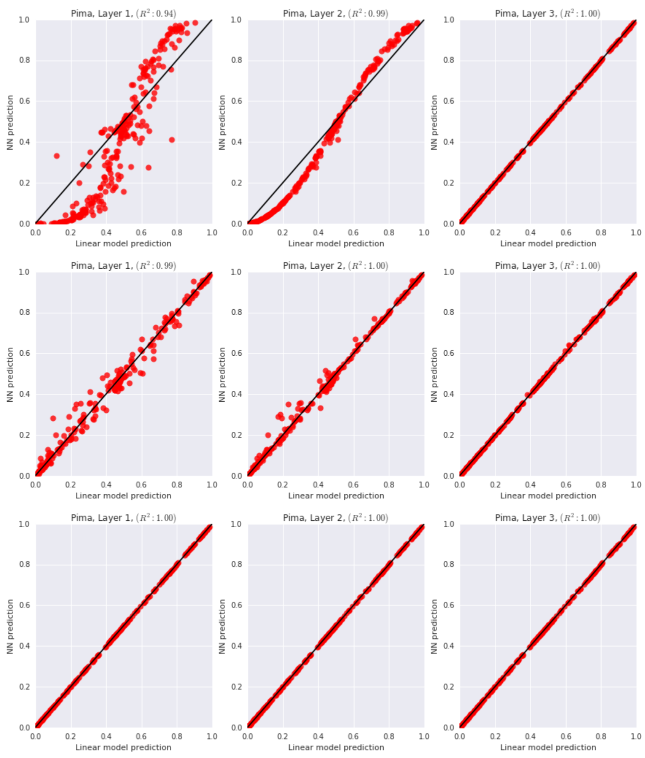

Figure 1: Classifier comparison for *Pima*. Training on activation features (row 1), holographic features (row 2), and closed form solution to holographic weights (row 3).

Model type	Pima	Census	MNIST-3v5
Neural net	5.9×10^{-5}	2.8×10^{-5}	1.1×10^{-4}
Activation	$< 10^{-15}$	$< 10^{-15}$	$< 10^{-15}$
Holographic	$< 10^{-15}$	$< 10^{-15}$	$< 10^{-15}$

Table 2: Average KL divergence between posteriors for different re-trainings of the same model on the same data.

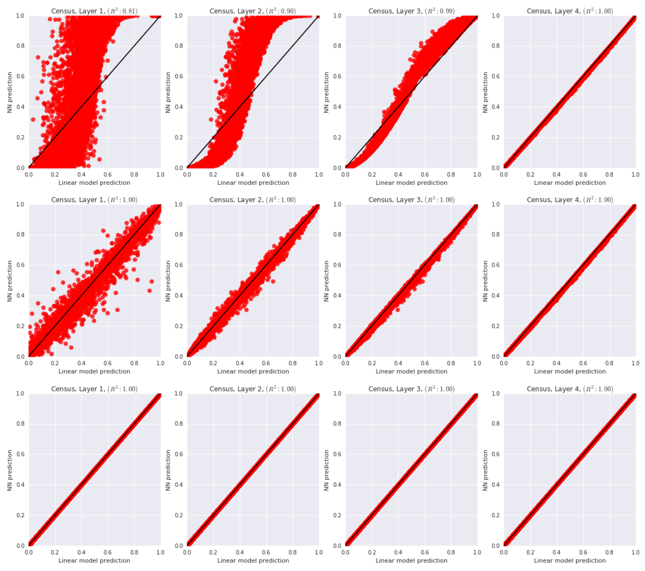

Figure 2: Classifier comparison for *Census*. Training on activation features (row 1), holographic features (row 2), and closed form solution to holographic weights (row 3).

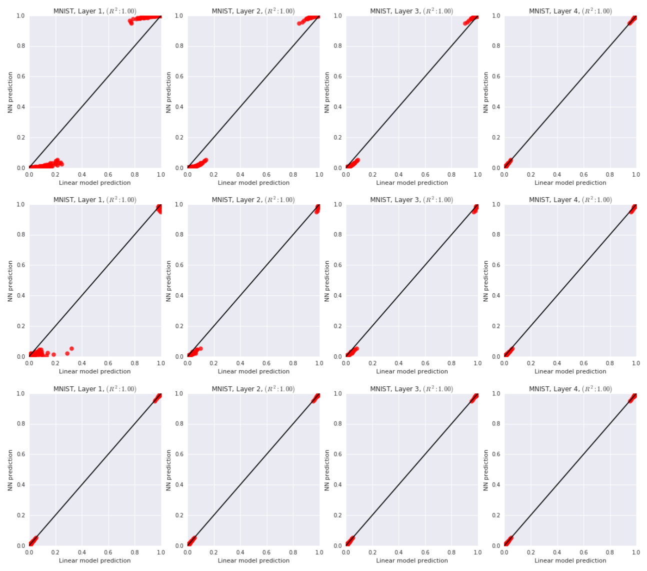

Figure 3: Classifier comparison for *MNIST*. Training on activation features (row 1), holographic features (row 2), and closed form solution to holographic weights (row 3).

	Pima		Census		MNIST-3v5	
Layer	Activ	Holo	Activ	Holo	Activ	Holo
1	.529	.462	.477	.331	.375	.133
2	.536	.455	.494	.339	.249	.085
3	.474	.471	.380	.345	.155	.082

Table 3: Average negative log-likelihood on third partition for linear classifier trained on second partition (after being extracted from first partition).

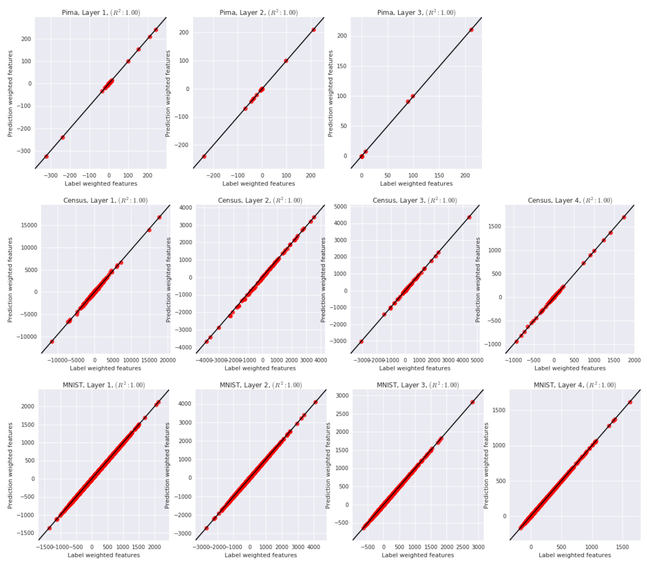

Figure 4: Calibration plots for extracted holographic features from each layer (columns) on each data set (rows).

which they were not inferred. In this case, we generated both holographic and activation features from a neural network that was trained on the first data partition, as above. Then we trained the linear models defined over these extracted feature sets on the second data partition, and assessed their negative log-likelihood on the third data partition. The results are given in Table 3. For comparison, the generalization performance of the neural network is Pima: 0.421, Census: 0.360, and MNIST-3v5: 0.086 (smaller is better). It is clear that the holographic features support better generalization than activation features, while achieving competitive generalization to the original neural network on the larger data sets.

10 CONCLUSION

We have introduced the concept of generated ("holographic") features from a model family, specified by the gradient of the prediction with respect to an appropriate subset of parameters. We have shown that, for any model family, if a model is at a local minimum, then the model is calibrated with respect to features generated from the parameters with respect to the model. Moreover, we have shown that for many standard feedforward networks, an optimal linear model on the generated features will make the same predictions as the original feedforward network. We show that in practice, building linear models with the generated "holographic" features better replicates the original network than building linear models based on intermediate activations.

References

[1] G. Alain and Y. Bengio. Understanding intermediate layers using linear classifier probes. *arXiv e-prints*, abs/1610.01644, October 2016.

[2] L. D. Brown. Fundamentals of statistical exponential families with applications in statistical decision theory. *Lecture Notes-Monograph Series*, 9:i–279, 1986.

[3] Y. Dauphin, R. Pascanu, Ç. Gülçehre, K. Cho, S. Ganguli, and Y. Bengio. Identifying and attacking the saddle point problem in high-dimensional non-convex optimization. *CoRR*, abs/1406.2572, 2014.

[4] D. Erhan, Y. Bengio, A. A. Courville, and P. Vincent. Visualizing higher-layer features of a deep network. Technical report, University of Montreal, 2009.

[5] D. Erhan, P.-A. Manzagol, Y. Bengio, S. Bengio, and P. Vincent. The difficulty of training deep architectures and the effect of unsupervised pre-training. In *AISTATS*, pages 153–160, 2009.

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

[7] A. Graves, A. Mohamed, and G. E. Hinton. Speech recognition with deep recurrent neural networks. In *ICASSP*, 2013.

[8] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

[9] J. Hirschberg and C. Manning. Advances in natural language processing. *Science*, 349(6245):261–266, Jul 2015.

[10] A. Krizhevsky, I. Sutskever, and G. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012.

[11] L. D. Kudryavtsev. Homogeneous function. In M. Hazewinkel, editor, *Encyclopedia of Mathematics*. Springer, 2001.

[12] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[13] A. L. Maas, A. Y. Hannun, and A. Y. Ng. Rectifier nonlinearities improve neural network acoustic models. In *ICML*, 2013.

[14] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In *NIPS*, 2013.

[15] A. Mordvintsev, C. Olah, and M. Tyka. Deepdream - a code example for visualizing neural networks. `https://research.googleblog.com/2015/06/inceptionism-going-deeper-into-neural.html`, July 2015.

[16] J. Nelder and R. Wedderburn. Generalized linear models. *Journal of the Royal Statistical Society Series A (General)*, 135(3):370–384, 1972.

[17] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015.

[18] D. Schuurmans and M. Zinkevich. Deep learning games. In *NIPS*, 2016.

[19] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun. Overfeat: Integrated recognition, localization and detection using convolutional networks. In *ICLR*, 2014.

[20] K. Simonyan, A. Vedaldi, and A. Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. *CoRR*, abs/1312.6034, 2013.

[21] M. J. Wainwright and M. I. Jordan. Foundations and trends®in machine learning. 1(1–2):1–305, 2008.

[22] Y. Wu, M. Schuster, Z. Chen, Q. V. Le, M. Norouzi, W. Macherey, M. Krikun, Y. Cao, Q. Gao, K. Macherey, J. Klingner, A. Shah, M. Johnson, X. Liu, . Kaiser, S. Gouws, Y. Kato, T. Kudo, H. Kazawa, K. Stevens, G. Kurian, N. Patil, W. Wang, C. Young, J. Smith, J. Riesa, A. Rudnick, O. Vinyals, G. Corrado, M. Hughes, and J. Dean. Google's neural machine translation system: Bridging the gap between human and machine translation. *CoRR*, abs/1609.08144, 2016.

[23] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson. How transferable are features in deep neural networks? In *NIPS*, pages 3320–3328, 2014.

[24] M. Yuan and Y. Lin. Model selection and estimation in regression with grouped variables. *Journal of the Royal Statistical Society B*, 68:49–67, 2006.

[25] M. Zeiler, G. Taylor, and R. Fergus. Adaptive deconvolutional networks for mid and high level feature learning. In *ICCV*, 2011.

[26] M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional neural networks. In *ECCV*, 2014.