Prediction Gradients for Feature Extraction and Analysis from Convolutional Neural Networks

Henry Z. Lo, Joseph Paul Cohen, and Wei Ding
Department of Computer Science, University of Massachusetts Boston, Massachusetts, United States

Abstract— **Despite their impact on computer vision and face recognition, the inner workings of deep convolutional neural networks (CNNs) have traditionally been regarded as uninterpretable. We demonstrate this to be false by proposing *prediction gradients* to understand how neural networks encode concepts into individual units. In constrast, existing efforts to understand convolutional nets focus on visualizing units and classes in pixel space, often using optimization. Our method for calculating prediction gradients is very efficient, and provides an effective technique to rank and quantify importance of internal units and their learned features based on the unit's relevance to any prediction. We use prediction gradients to analyse the features learned by a CNN on a standard face recognition data set. Our analysis identifies strong patterns of activation which are unique for each identity. In addition, we validate the rating produced by prediction gradients to remove the most important features of the network, knocking out their respective units in the network, and demonstrating detrimental effects on network prediction. Our experiments validate the utility of the prediction gradient in understanding the importance and relationships between units inside a convolutional neural network.**

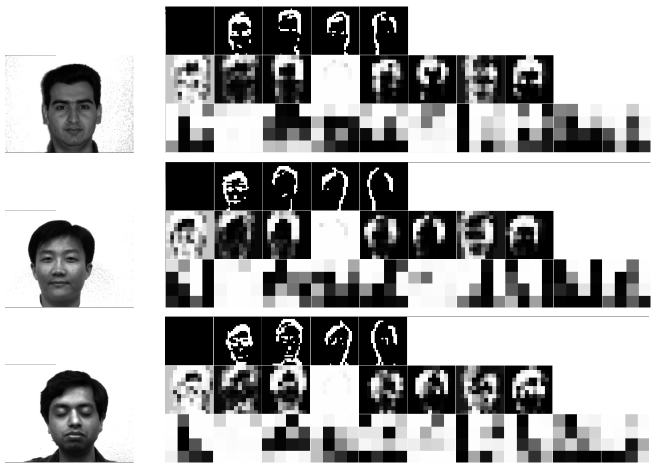


Fig. 1. Convolutional outputs for three images. For each face in the left column, the images in the right show the output of each convolutional filter. Rows represent layers. Note that some units have become unresponsive.

I. Introduction

Deep architectures have made great strides in image understanding. Krizhevksy's landmark performance on the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) validated convolutional neural networks (CNNs) in particular, leading to all subsequent submissions using variants of CNNs [?], [?]. This success is punctuated by the fact that ILSVRC consisted of millions of images and one thousand classes, and is perhaps the largest challenge of its kind [?]. These advances have since spread to many domains, including face recognition and detection [?], [?].

The CNN uses serially stacked generalizations and shared weights to drastically reduce the number of learned parameters [?]. The last layer of the network is typically a logistic regression or softmax layer, a linear classifier; that this works suggests that all other layers exist only to learn a useful feature vector for the classification task. This realization has led to representation learning becoming a synonym for deep learning [?].

Despite the crucial importance of these features, *how* these features are internally represented is widely regarded as uninterpretable. This "black box" nature of neural networks is one of the technique's common criticisms.

This work was supported by the Sanofi Genzyme Doctoral Research Fellowship, the College of Science and Mathematics Dean's Office at the University of Massachusetts Boston, and partially supported by the Department of Energy (# E2024282) and the National Science Foundation Graduate Research Fellowship Program (# DGE-1356104).

In fact these internal representations are interpretable. We demonstrate an efficient technique for evaluating the importance of features learned by units in a neural network by looking at activations. Using the prediction gradient, we are able to rank feature importance, not only overall, but also with respect to given classes.

Existing attempts to understand convolutional neural networks have focused on visualizing units, layers, and classes back into pixel space; for example, in Figure 1. This approach reveals little insight into how the neural network encodes the image as a function of its units. However, knowing the usefulness of units can help with understanding how concepts are distributed in a neural networks.

We show that the prediction gradient can be used to understand what happens inside a convolutional net in two ways. First, we use it to visualize activation patterns in a convolutional net, and show that similar classes often induce similar unit patterns. Second, we demonstrate the method's effectiveness in identifying features in a network. By removing identified 'strong' units, we show that performance degrades much faster than removing identified 'weak' units.

Our contributions are:

1) An efficient technique for measuring the importance of any unit in a deep neural network, using the *prediction gradient*, *class gradient*, and *convolutional gradient*.
2) With these gradients, a demonstration that different images of the same class show related activation patterns.
3) Extraction of strongest and weakest features using the

Symbol	Meaning
o_a^ℓ	Output of ath unit in ℓth layer
$\mathbf{y}^j$	Output vector of neural network for jth input
w_{ab}^ℓ	Weight from ath unit in $(\ell-1)$th layer to the bth unit in the ℓth layer
t^i	Identity of ith image
X_t	Set of all images sharing the label t
L	Loss function

TABLE I
NOTATION USED IN THIS PAPER. SUPERSCRIPTS AND SUBSCRIPTS MAY BE OMITTED WHEN ONLY CONSIDERING ONE COPY OF A SYMBOL.

proposed method, which is validated by their direct effect on network performance.

II. Related Work

Existing work seeks to understand convolutional nets in terms of pixel space. To our knowledge this is the first paper to investigate how to understand convolutional network concepts in terms of its units.

Erhan et al. visualized convolutional neural nets by finding the pixels in input space which maximize the activation of a given unit [?]. The authors also propose a sampling method from deep belief networks.

Zeiler and Fergus visualized the concepts learned in each unit of a network using deconvolutional net [?]. These networks are optimized to reproduce the input given a unit's activation [?].

Simonyan et. al. visualized object classes by backpropagating class predictions back to the input pixel space. They also proposed a method for identifying the pixels of a given image relevant to its object class [?].

Donahue et. al. compared several different dimensionality reduction methods on the high dimensional feature space generated at any given layer of a deep network, and proposed a method which selects image segments from pixel space to visualize the network [?].

Erhan and Zeiler's methods both map a single unit back to pixel space. Simonyan's methods visualize object classes, also in pixel space. Our proposed method seeks to understand object classes in terms of units in the convolutional network.

We propose a fast method for ranking and evaluating units in a neural net, which requires no additional optimization. By providing a single number for each unit, our method quantitivately measures each unit's contribution to a class prediction. In contrast, existing methods only visualize the contribution of a single unit by mapping back to pixel space, and several require nonconvex optimization. Furthermore, our measure is flexible, as it allows unit / feature importance to be calculated with respect to a given class input, prediction, or overall.

III. Prediction Gradient

The notation used in this paper is shown in Table I.

A. Prediction Gradient

We use the *prediction gradient* to quantify a unit's relevance for a given prediction:

$$\frac{\partial \mathbf{y}}{\partial o} = \left\langle \frac{\partial y_1}{\partial o}, \ldots, \frac{\partial y_{|T|}}{\partial o} \right\rangle \tag{1}$$

This vector measures how much the unit o contributes to the final output $\mathbf{y}$ for a given image. The ith element of the prediction gradient refers to the change in predicting class i due to varying o.

In our analyses, we focus what makes the network predict correctly, so we investigate the element y_i where $i = t$ is the face label. The other elements of the prediction gradient provide insight into how unit o contributes to misclassifying the image. This may be of interest in understanding why the network makes certain mistakes.

B. Calculation

One major advantage of using the prediction gradient is that it can be calculated very quickly. We will show that prediction gradients are already calculated as an intermediate step during backpropagation. Thus, obtaining all the prediction gradients of a dataset is relatively simple, and no slower than one epoch of training.

Let L be an loss function which depends on $\mathbf{y}$. Given one image, online backpropagation will update weight w_{ab}^ℓ using this equation:

$$w_{ab}^\ell = w_{ab}^\ell + \lambda \frac{\partial L}{\partial w_{ab}^\ell}$$

The weight w_{ab}^ℓ feeds into unit o_b^ℓ, and thus the error attributable to o_b^ℓ must be calculated before the error attributable to w_{ab}^ℓ can be determined. Mathematically, this becomes apparent after applying the chain rule:

$$\frac{\partial L}{\partial w_{ab}^\ell} = \frac{\partial L}{\partial \mathbf{y}} \frac{\partial \mathbf{y}}{\partial o_b^\ell} \frac{\partial o_b^\ell}{\partial w_{ab}^\ell} \tag{2}$$

Thus, the prediction gradient is already calculated in the course of learning, and requires no additional effort to obtain.

C. Class Gradient

To quantify unit relevance to each class label, we use the *class gradient*, which is the average prediction gradient over all images of the same class t:

$$\frac{1}{|X_i|} \sum_{j \in X_i} \frac{\partial \mathbf{y}^j}{\partial o} = \frac{1}{|X_i|} \frac{\partial}{\partial o} \sum_{j \in X_i} \left\langle y_1^j, \ldots, y_{n_c}^j \right\rangle \tag{3}$$

Equation 3 shows that the average prediction gradient is equivalent to the gradient of the average prediction. In practice, we found that calculating the class gradient in this way is easier to parallelize.

For some datasets, averaging may be problematic if the elements of a class are not sufficiently smooth, that is, if different images of the same class have very different activation patterns. We show later that this is not the case for our data set.

The class gradient is useful in understanding how the neural network perceives all images of each class, rather than how it predicts one specific image.

D. Convolutional Gradient

In this paper, we focus on the class gradients of the convolutional layers in a convolutional neural network. Each layer can be thought of as a mode-4 tensor W^ℓ:

- Mode 1 corresponds to the different feature maps.
- Mode 2 corresponds to the different channels of the layer's inputs (e.g. one channel for each convolutional output in the low layer).
- Mode 3 and 4 correspond to the 2-d convolutional filter location in the image.

Sharing this weight tensor W^ℓ are many units: the amount is the number of valid input pixels times the number of feature maps. All units of a feature map arguably correspond to a single feature, so we quantify the importance of this entire set of units by aggregating all units in a map using the Frobenius norm. We call this the *convolutional gradient*. δ_a^ℓ is the ath convolutional gradient in the ℓth layer.

By aggregating, the convolutional gradient removes spatial information in a feature map. However, as we want to quantify the importance of a feature map, there is no way to avoid this. If we wanted to take into account spatial information, we could use prediction or class gradients instead.

The convolutional gradient is not a true gradient, and so is not directly comparable to the non-convolutional prediction gradients. However, it is a useful quantity to compare among feature maps.

IV. Experimental Setup

We use the gradients discussed to investigate an intentionally small network and data set to aid interpretability of results. Using a learned convolutional network, we do the following:

1) Identifying activation patterns relevant to correct class prediction, and show that these are relatively stable for different images in the same class.
2) Ranking and rating units in a neural network, to observe how the network understands face identities.
3) Validation of the measure of unit contribution by removing the strongest / weakest features, and observing the effect on performance.

Our goal is to demonstrate a measure of relevance of each unit to correct prediction. Our activation patterns, feature rankings, and knockout experiments support one use of the gradients, but there may be others. The purpose of the analyses is to demonstrate the utility of the class gradient, and hence we use a simple data set and neural network to maximize interpretability.

A. Model

We use a convolutional neural network similar to LeNet [?], but with the following parameters:

- 3 convolutional layers, with 5, 8, and 10 feature maps.
- One hidden layer with 20 units.
- Hyperbolic tangent activation functions.
- 5×5 convolutional filters for all convolutional layers.
- 2×2 max pooling for all convolutional layers.

Input images are resized to 60×60 pixels. The learning parameter is set to 0.1.

B. Data

To facilitate learning on our network, we use the Yale face database[1], which consists of face images of 15 identities in 11 different conditions [?]. 75% of the data set is used for training, and 25% for testing. The network was trained on 15 classes, one for each identity.

Training was done for 250 epochs, after which the network achieved an error rate of 4.88% on the data set. Weights for the network were frozen after this, and no more training was done.

V. Gradient Analysis

We demonstrate the utility of the prediction gradient by example. Using on a CNN trained on a face recognition data set, we show how the prediction gradient can be used to understand the inner workings of a neural network, and to identify useful and non-useful features relevant to class labels.

A. Prediction Gradient Consistency

In order to use the class gradient for analysis, prediction gradient patterns must be similar among members of the same class and different between classes. Otherwise, averaging these over the class would be meaningless.

In Table II, we conducted a preliminary investigation to see whether this condition is fulfilled. Shown are three different identities, under three different conditions. To the right of each image are the convolutional gradients for each convolutional feature map. The whiter the circle, the higher the convolutional gradient for that feature map. Black corresponds to a gradient of 0.

We can see that which units have the highest gradients stays quite similar across different faces of the same identity. In other words, each person has a distinct gradient profile of which units contribute most to detecting that face.

B. Gradient-Based Feature Selection

The convolutional gradient can be used as a measure of a unit's effectiveness in producing a certain network outcome. Hence, it can be used rank the internal features of a neural network in terms of importance.

In Table III, we use the convolutional gradient to identify the top 8 strongest and weakest features for each of five identities. We average these features across different images of the same identity. Note that as the gradients are different for each identity, each identity has a different set of strongest and weakest features.

The values for these convolutional gradients are shown, and all convolutional gradients are visualized in the network diagram in the third column. We note that there is a high variability in the value of the gradient. For example, identity 15's gradients are much higher than identity 14's. Thus, in our network visualizations, we normalize based on the highest gradient in the network.

[1]http://vision.ucsd.edu/content/yale-face-database

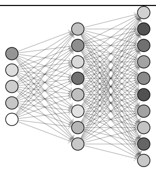
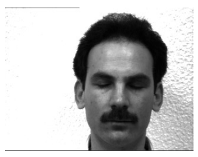
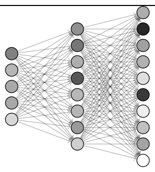
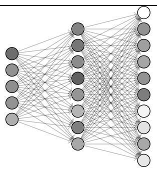
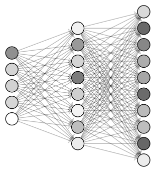
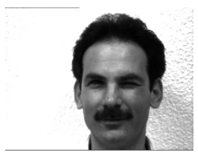
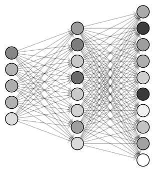
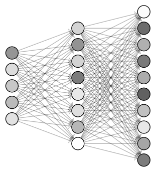
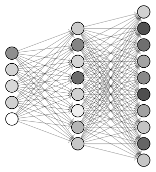
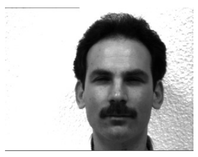
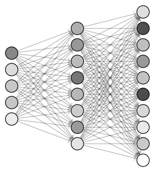
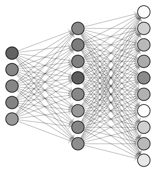
Identity 1		Identity 2		Identity 3	
Image	Network	Image	Network	Image	Network
					
					
					

TABLE II
PREDICTION GRADIENTS FOR THREE IMAGES EACH FROM THREE IDENTITIES. THE DIAGRAM IN THE NETWORK COLUMN THE THREE CONVOLUTIONAL LAYERS OF THE NEURAL NETWORK. EACH CIRCLE IS A FEATURE MAP. BRIGHTER CIRCLES INDICATE A HIGHER PREDICTION GRADIENT FOR THAT FEATURE MAP. NOTE THE PATTERNS IN PREDICTION GRADIENTS WITHIN EACH IDENTITY.

C. *Gradient-Selected Feature Evaluation*

To see whether the features ranked by the network are indeed the most and least useful, we *knock out* these units in the neural network and see its effect on network performance.

Specifically, for each identity (we use identities 7, 10, 12, 14, and 15), we find their top 10 strongest features. Then for each of these features, we set the output of the corresponding unit corresponding to 0 (knocking it out). We then run the modified network to evaluate its performance. The effects of the knockout are cumulative; e.g. at the first iteration we knock out the best feature, then we knock out the best 2 features, etc.

We evaluate network performance in two ways. First, we measure class error by only testing the network on the identity whose strongest features were extracted. Second, we measure overall error by testing the network on all identities.

Finally, we do this same experimental procedure for the 20 weakest features. We use the entire Yale data set, not just the test set.

D. *Gradient-Selected Feature Results*

Knockout experiment results are shown in Figure 2.

It is evident that removing three or more of the strongest features strongly impact network performance. There seems to be some robustness to the network, in that it can handle minor damage and still classify well. This robustness depends on the identity; e.g. the network fails to recognize identity 12 after removing 4 units, but only fails on identity 10 after removing 8.

Performance on recognizing all identities seems to degrade at a slower rate that recognizing a single identity, at least for the top features for identities 10 and 15. Regardless, all of the strongest feature near 100% error after removing 10.

Removing the weakest units seems to have an odd effect on class performance. For most of the weakest feature sets, error gradually increased as features were knocked out, but the pattern is not certain. For example, error for identity 12 increased drastically after removing the 7 weakest features, but then dropped again after removing two more. This suggests that some of the weak features, or sets of weak features, actually hurt network performance.

Overall performance degrades very slowly when removing features from the weakest first. The network almost completely fails after removing the top 10 strong features for any identity. In contrast, after removing 10 weak features, the network still achieves less than 25% error. Even after removing 15 weak features, overall error for many of the weak feature sets is below 50%. This is remarkable, considering that there are only 23 convolutional features.

Misclassifications for the knockout procedure can be seen in Figure 3. The strongest features for identity 7 were used. The colored lines in this image show the true class label, while the rows show the mode predicted class label. Interestingly, 7 is not the first identity to be grossly misclassified. As the network degrades, identities become more difficult for the network to distinguish.

E. *Identifying Problematic Units*

The performance patterns in Figure 2 suggests that weak features for a given class are often weak overall. These units may have become saturated, or fit to noise [?], as often happens with deep networks. Irrelevant features often hurt accuracy [?], [?], so it is important to identify units as they arrive at this state.

From the convolutional images in Figure 1, it seems that feature 1 in layer 1, feature 4 in layer 2, and feature 2 in

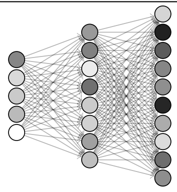
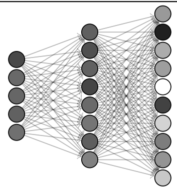
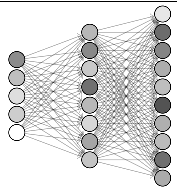
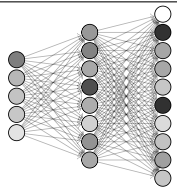
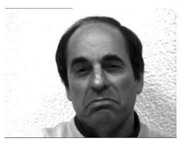
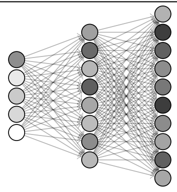
ID	Image	Network	Strongest Features			Weakest Features		
			Layer	Unit	Gradient	Layer	Unit	Gradient
15			1	5	0.25153	3	2	0.03227
			2	3	0.23269	3	6	0.03676
			3	8	0.21675	3	3	0.09039
			1	2	0.21362	3	9	0.10847
			3	1	0.20919	2	4	0.10995
			2	6	0.20612	2	2	0.12622
			2	5	0.19961	3	4	0.13164
			1	3	0.19855	1	1	0.13169
14			3	5	0.00202	3	2	0.00025
			3	7	0.00167	3	6	0.00052
			3	10	0.00158	2	4	0.00055
			3	3	0.00136	1	1	0.00057
			3	4	0.00125	2	2	0.00063
			3	1	0.00121	2	7	0.00075
			3	9	0.00117	2	1	0.00076
			2	8	0.00102	2	3	0.00077
7			1	5	0.02247	3	6	0.00728
			3	1	0.02029	3	2	0.00951
			1	3	0.01951	3	9	0.00977
			2	6	0.01890	2	4	0.00983
			1	4	0.01802	3	3	0.01157
			2	3	0.01760	2	2	0.01199
			2	8	0.01726	1	1	0.01228
			1	2	0.01683	2	7	0.01453
10			3	1	0.00369	3	6	0.00071
			1	5	0.00328	3	2	0.00072
			3	7	0.00316	2	4	0.00114
			2	6	0.00301	1	1	0.00182
			1	4	0.00287	2	2	0.00189
			3	5	0.00287	2	7	0.00212
			3	10	0.00286	2	1	0.00219
			1	3	0.00282	3	9	0.00231
12			1	5	0.08089	3	6	0.01925
			1	2	0.07403	3	2	0.01964
			1	4	0.06849	2	4	0.03013
			1	3	0.06410	3	9	0.03074
			2	6	0.05959	3	3	0.03077
			2	8	0.05814	2	2	0.03345
			2	3	0.05761	3	5	0.03780
			3	1	0.05623	3	7	0.04417

TABLE III

FIVE IDENTITIES WITH THEIR IMAGES, CLASS GRADIENTS ACROSS THE WHOLE NETWORK, AND TOP 8 STRONGEST AND WEAKEST FEATURES, AS RANKED BY THE CLASS GRADIENT. THESE FIVE IDENTITIES WERE USED IN THE KNOCKOUT EXPERIMENTS.

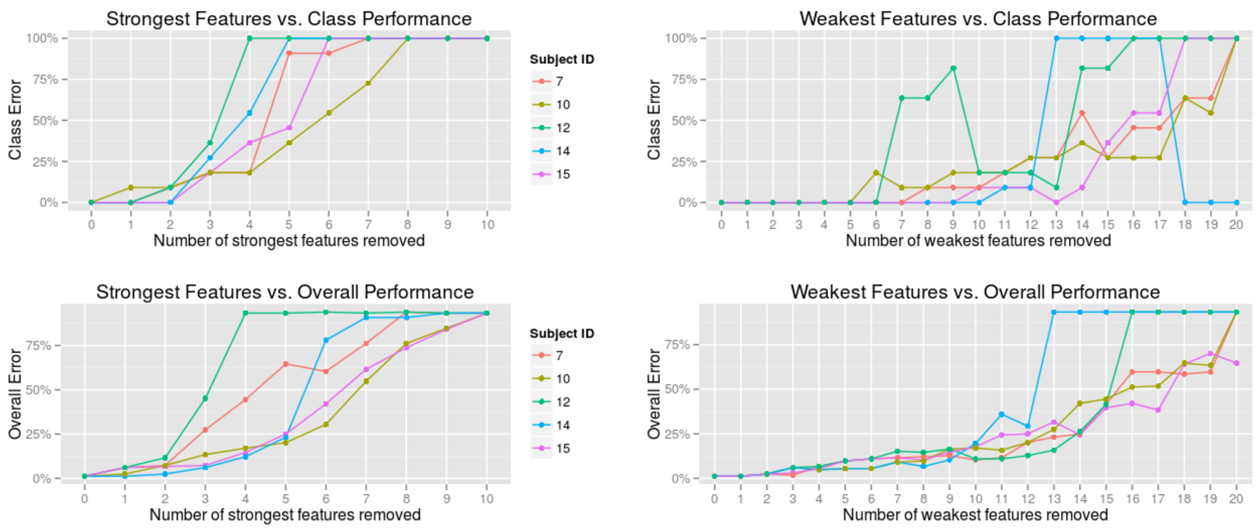


Fig. 2. Charts show the effect of removing an identity's strongest features on network performance. Two types of error are considered - error within the identity, and error among all images. Left column shows the removal of strongest features one-by-one, and right column shows the removal of weakest features. Features are rated based on class gradient.

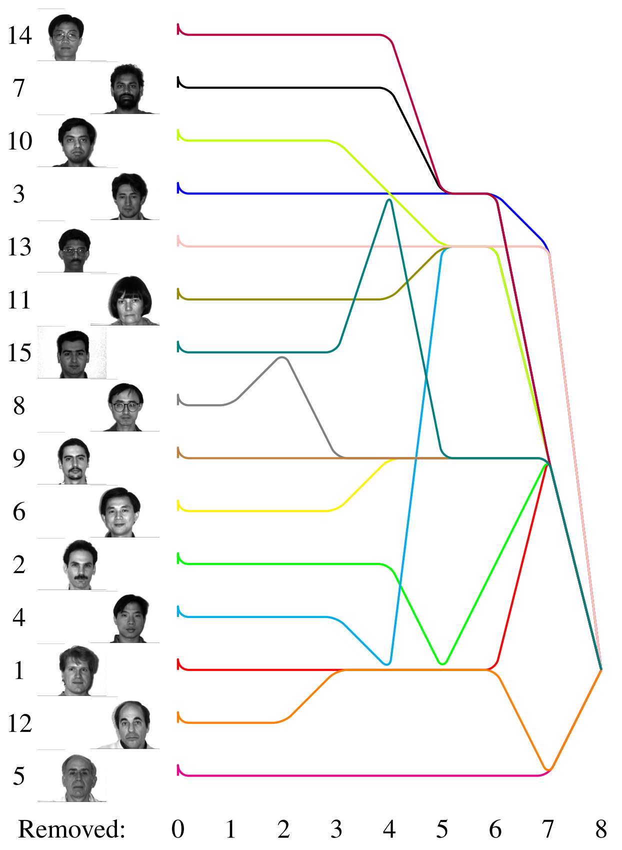


Fig. 3. Visualization of how the neural network misclassifies identities as the strongest features for identity 7 are knocked out. Lines of different colors represent subsets of the data representing one identity. A line in a given row means that the images of the line are most often predicted to be the identity in the row. At the left, all subjects are correctly classified as no features are removed yet. As the lines move right, units are knocked out, and the ability to distinguish between faces is lost.

layer 3 do not seem to detect much. However, it is not safe to assume this based purely on visualizations.

The low prediction gradients for these features as shown in Table III confirm that indeed these features have become irrelevant. In addition to these low-performers, feature 6 in layer 3, feature 9 in layer 3, and feature 2 in layer 2 are also low in relevance, though this may not be visually apparent. Using the class gradient to determine relevance adds certainty to the judgement of relevance.

VI. Conclusion

Utilizing the convolutional gradient, we have extensively analyzed a convolutional neural network trained on a standard face recognition data set. The analysis revealed the following facts about the neural network:

- Stability of representation. Each identity learned by the neural network corresponds to a sparse activation pattern, which forms a ”profile” for each given identity.
- Robustness of representation. Even when removing one or two of the strongest features in a neural network, its robust internal representation still allows for high-performance.
- Feature relevance imbalance. As shown in Figure 2, even after removing half of a weak feature set performance remained high. Yet removing more than 3 strong features resulted in severe performance degradation.

This analysis was made possible using prediction gradients and knockout analysis. Prediction gradients have the benefit of being efficiently computable, unlike modern methods for understanding neural networks, yet non-linear, unlike traditional measures such as correlation. This method is applicable not only to convolutional layers as demonstrated in this paper, but to any output in a neural network, and thus can yield deep insights into the deep neural network architectures which have revolutionized machine learning and computer vision.