XNET: TOWARDS TRAINING EXPLAINABLE NEURAL NETWORKS

Kurtis Evan David
Department of Computer Science
University of Texas at Austin
{kurtis.david}@utexas.edu

Qiang Liu
Department of Computer Science
University of Texas at Austin
{lqiang}@cs.utexas.edu

ABSTRACT

Explainability in deep learning has been a thoroughly discussed topic and many solutions have provided ways to analyze information captured during training. These do not guarantee inherently interpretable visualizations; as a solution to this, we propose XNet: a network that utilizes a novel regularization term that drops neurons during training. The hope is that by removing a certain neuron, we can induce a large classification loss, because this neuron has captured an important feature about a conditioned category. We investigate how to make this work through several hypotheses and show our results on the CelebA dataset. Finally, future work still must be done in visualizing what our new network has learned – current post-hoc techniques do not make this clear yet.

1 INTRODUCTION

Explainability in machine learning has become an important topic as the public and media become more aware of the "black-box" nature of state of the art algorithms. This is especially true of neural networks, which can learn highly complex functions but can be disregarded due to a lack of intuitive explanations compared to general additive models and decision trees. Much work has been placed on *post-hoc* analysis of networks, where models are explained *after* training, in hope that something reasonable comes out.

One such method analyzes a trained network by backprojecting the influence of pixels in the image space w.r.t. network activations. Simonyan et al. (2013) are one of the first to do this, where an influence of a pixel is defined as its gradient for the correct class activation. These gradients can be naturally visualized on pixel space and are referred to as saliency maps. Bach et al. (2015) improve this framework by allowing positive and negative influences, as well as connecting the theory to Taylor series. Zhou et al. (2016) move away from saliency maps and instead create heatmaps based off of final convolutional activations from a global activation pooling layer. These tend to be easier to understand visually and provide different heatmaps for each class.

Other work instead focus on the topic of this paper – making neural networks more explaining *during* training. Zhang et al. (2017) make convolutional filters more explainable by restricting their receptive fields to individual object parts. They bypass the use of additional part labels by incorporating a prior that high filter activations should be centered at some pixel and lessen away from the center. Li et al. (2017) create an entirely new type of layer, a Prototype layer, that performs a deep nearest neighbors in the latent space of an autoencoder – distances away from prototypes in this space are used as features for classification.

In each of these works, neuron activations are not explicitly conditioned on auxiliary data, e.g. annotated attributes or categories. The goal of this project is to do just that – given that an image contains a certain attribute, can we force a neural network to activate a particular neuron? We would like to create a whole fully connected layer of these explainable neurons, each conditioned on different categories. This would then create a natural explanation for the model; the basis dimensions of this layer (i.e. each neuron) can now be explained through known semantic attributes.

2 PROPOSED METHOD - DROPPING NEURONS

To create this explainable network, we first make the following observation: if a neuron is *important* to a certain category of images, then the neuron's activation on this subset must be significant. i.e. If this neuron were to be "dropped" from the input to the next layer, then the classification loss should for these examples should (drastically) increase. In other words, the neuron has captured something about the category that is essential to a correct classification. To formally define this significance, let us first set some notatation.

$$\begin{aligned}
\mathcal{L} &\triangleq \text{Classification loss} \\
\mathcal{X} &\triangleq \text{Minibatch of Images} \\
\mathcal{X}_j &\triangleq \text{Images within Category } j \\
\theta &\triangleq \text{Network parameters} \\
\hat{\theta}_j &\triangleq \text{Network parameters without neuron } j
\end{aligned}$$

The aforementioned observation can now be described as the following optimization objective:

$$\min_{\theta} \mathcal{L}(\theta, \mathcal{X}) - \alpha \sum_{j=1}^{k} \mathcal{L}(\hat{\theta}_j, \mathcal{X}_j) \tag{1}$$

This assumes that neuron j will capture information about category j, and that there exist k categories in all. (1) is just an augmented loss function with an α regularization term, which we will refer to as the *drop loss*, that encourages the network to perform *worse* after neuron j has been disconnected from the network – see Figure 1 for visual details.

Thus, to train our explainable network, Equation (1) just needs to be minimized, and can be done through auto-differentiation libraries such as PyTorch (Paszke et al., 2017). However, through the course of this project, this minimization did not work as expected; through many tests, we have come up with Equation (2) – we would like to maximize the *minimum* of all drop losses within a minibatch.

$$\min_{\theta} \mathcal{L}(\theta, \mathcal{X}) - \alpha \min_{j} \mathcal{L}(\hat{\theta}_j, \mathcal{X}_j) \tag{2}$$

We do this, because a simple sum or mean can just maximize the drop loss of a few select neurons, while the others are not conditioned at all. We go into more detail with this issue in Section 4, and provide other solutions tested.

2.1 EXPERIMENT SETUP

To test our defined regularization, we use the CelebA dataset (Liu et al., 2015) and utilize the Young/Old label to act as our binary classification task. Our architecture consists of convolutional layers from ResNet-18 (He et al., 2015); after the global average pooling layer, we include the explainable fully connected layer with 3 hidden units, then the final softmax output layer. As part of our framework, we condition each neuron, respectively, in the hidden layer with the following auxiliary categories: 5 O'Clock Shadow, Bald, and Bangs.

The baseline architecture for comparison is the exact same, except we set $\alpha = 0$. We just want to observe the training dynamics of the same setup without any regularization and see if any noticeable differences arise. We compare the classification accuracy of both models, as well as the associated drop losses of the network. In addition, we provide other ways of visualizing the effect of our regularization, which can be seen in Section 3.

2.2 IMPLEMENTATION DETAILS

We first want to provide specific details on how we implement the idea of "dropping" neurons. Based on our notation, let j denote the desired neuron to be dropped. To ensure that only $\hat{\theta}_j$ is optimized,

we create an *almost identity* layer, the X-layer, which performs the operation. The X-layer is a simple fully connected layer with weights set constant to be the identity I_k. Given a certain neuron j to be dropped, we simply zero out the j^{th} component on its diagonal. This ensures that no gradient will propagate backwards towards or from neuron j, guaranteeing we only optimize for $\hat{\theta}_j$.

In addition, for reproducing capabilities, we set the following hyperparameters. We use SGD with an initial learning rate of 0.1 and momentum 0.9, and reduce the learning rate every two epochs by a factor of 10; a batch size of 128 was used as well. We found $\alpha = 0.05$ to be suitable, and apply early stopping after the eighth epoch through the dataset. Any further training resulted in a massive dip in both classification accuracy as well as the desired drop losses.

3 Results and Visualizations

We first want to check the test accuracy between our baseline architecture and XNet. This can be found in Table 1. We see that XNet has considerably worse test accuracy, possibly due to the additional regularization term that does not improve generalization. Note that we are trying to aim for more *explainable* networks, and this does not equate to networks that generalize better. Another possible explanation for the reduced accuracy is the fact that the drop loss constrains neurons to capture information about certain categories, which may not necessarily be useful for the final classification. One clear example of this is 5 O'Clock Shadow, as both young and old people can have this feature. Future work will cover possible ways to improve this aspect.

Architecture	**Test Accuracy**
Baseline	0.8817
XNet	0.8558

Table 1: Test Accuracy on Predicting Young/Old.

As a second means of comparison, Figure 1 contains drop losses of each individual neuron in the two architectures. The trend that we should see is that as training progresses, the drop losses for each neuron should consistently increase. This is indeed the case for the XNet training, compared to the baseline, where it seems that a single neuron has a high drop loss. In addition, this single neuron may have a high drop loss only because of initialization – this is evidenced by the fact that Neuron 1 in both graphs start at the highest drop loss. An interesting thing to note is the fact that the baseline contains such a significant neuron, even without conditioning, and it may be promising to study this further. e.g. Can we make all drop losses small to possibly increase robustness in the network?

Another way to verify the significance of a category for a neuron is to collect activation values for each neuron for every category. The intuition here is that a significant neuron should activate the most under the category it is conditioned on. Thus, to do this, we record category specific activations for every neuron, and normalize under total sum of its activations. A comparison of this analysis between the two training schemes can be seen in Figure 2. Although in each case categories maximally activate unique neurons, we see that XNet has more specialized neurons, since the highest category activations are significantly higher than the rest.

3.1 Interpretability

Although we have shown that the neurons have been specialized for their target categories, we still would like to see if interpretability of the network has improved. To do this, we implement three methods of post-hoc analysis for neural networks:

1. Saliency maps. We take sample images from each category and compute their corresponding maps. (Figure 3)
2. Class activation maps. Similar to the above method, we overlay these heatmaps onto images from each category. (Figure 4)

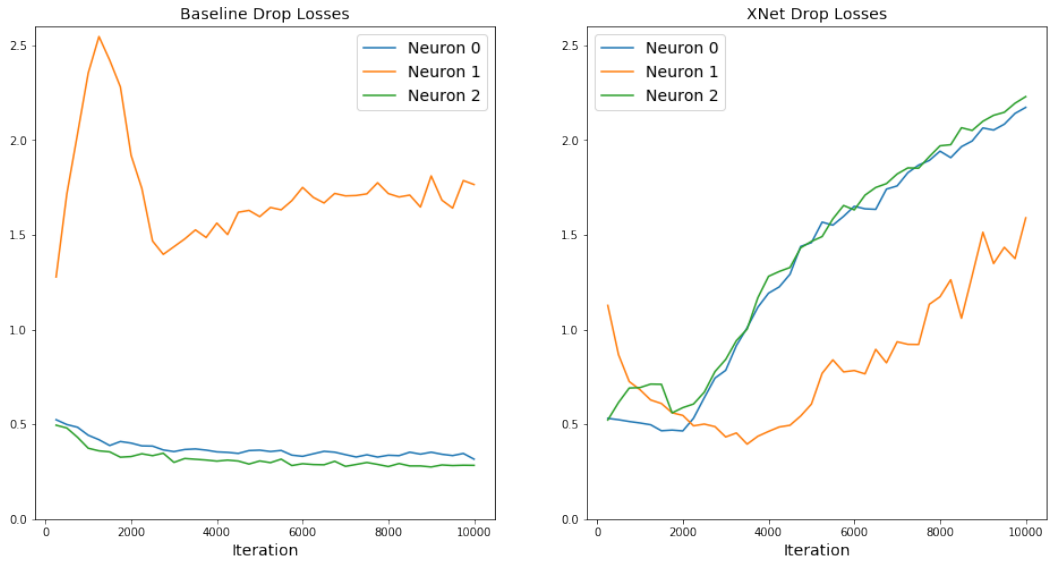

Figure 1: Drop Losses for each neuron in a fully connected layer. We see that without any conditioning, only one neuron tends to be very sensitive, whereas XNet is able to induce this for every neuron.

3. Visualizing Top-9 Images with Highest Drop Loss for each neuron. This analyzes the network at the neuron level, and the hope is we see the the same categories that are conditioned in XNet. (Figure 5 and 6)

For Figure 3, we can judge that the saliency maps are not very convincing for the positive examples – we only see slight activations in the general area that they should be activating in. Additionally, these were already cherry-picked; on the other hand, in the right group, we can clearly see that XNet saliencies focus on completely different features, mainly near the center of the face and forehead. Something interesting to note is that sometimes the baseline network creates more interesting saliencies; fore example, in the second row left group, we see the baseline map activate highly on the 5 O'Clock Shadow, whereas XNet does not. To see possibly better results, we turn to class activation maps.

In Figure 4, we see more convincing positive results, but equally strong negative results. For 5 O'Clock shadow, it focuses on a main area of stubble near the chin area; however, in the negative example, it concentrates on the central area of the face. This aligns with what we notice in Figure 3, as well as the negative example for the Bald category. The positive example for bald hones in on the bald forehead area, as well as the bangs category. However, this should be taken with a grain of salt, because the Drop Loss for the bangs example is 0.04 – the regularization term does not even correctly recognize the bangs within the image. Overall, these are easier to visualize and vary greatly between the baseline and XNet models; however can still be weak indicators of the interpretability of our new network.

Lastly, we find the top 9 images that induce the highest drop loss for each neuron in the explainable layer of XNet. This is similar to Szegedy et al. (2014) where they visualize basis directions (neuron activations) by maximally activating a neuron, however we focus on the final drop loss only. Compared to the past two methods, this seems the most promising, as we see clear similarities in each group in Figure 5. This is in contrast to Figure 6, where the only group that has convincing similarities would be the right-most group, that seems to have captured curly hair.

The first neuron should be capturing 5 O'Clock shadow, and there exist a few examples in this group that exhibit this category. The ones that do not though tend to have sharp/chiseled jaw/chin areas, so this neuron may have picked up on this pattern as well. The second neuron, on the other hand, should have picked up the bald category, but also has clearly picked up on African-Americans

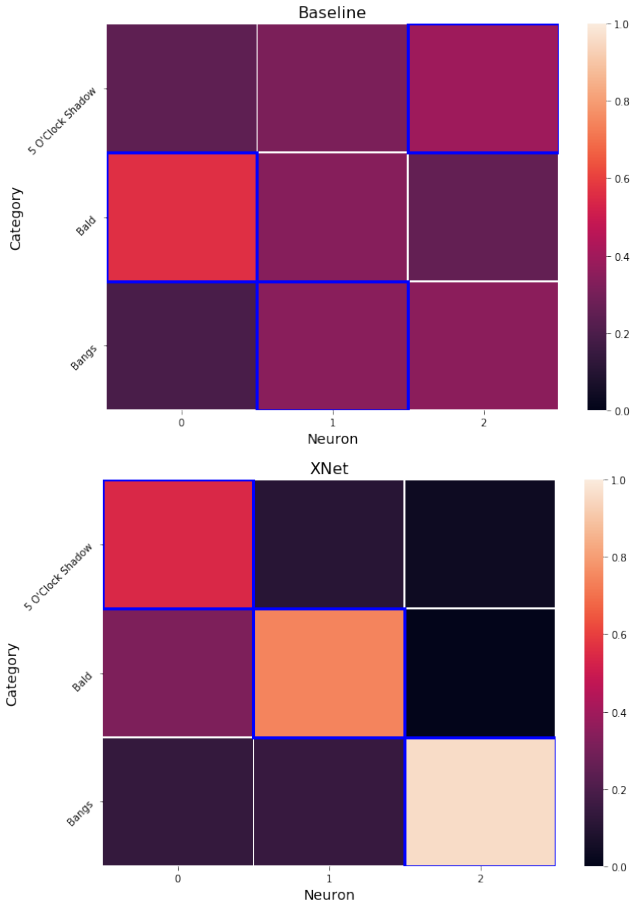

Figure 2: **Top:** Activation heatmap of baseline fully connected layer. **Bottom:** Activation heatmap of XNet fully connected layer. Both are normalized over neurons (columns). Blue squares indicate highest value in each column. We see that XNet has more specialized neurons.

within the dataset. This is most likely due to a correlation in the test set, in that the African-American celebrities tend to be bald.

Now the final neuron has actually picked up on the category it was conditioned on – bangs. Its quite convincing, as every photo in the top 9 has bangs. Although this does not complete prove/explain what information the neuron has learned to pick up on, it is a step in the right direction in trying to visualize the effects of XNet.

4 Challenges and Tested Hypotheses

In this section, we would like to cover a few solutions that we tried in order to get the framework to work. As mentioned in Section 2, we want to first cover the initial issue with Equation (1). By optimizing the sum or mean of drop losses, the network can easily focus on a single neuron, as it was already doing in baseline training (since this would increase the overall sum/mean). This indeed is the case when using (1) as the optimization objective, and this can be seen in Figure 7. However, we want *all* neurons to increase in drop loss, not just a select few – thus we chose to optimize for the *minimum* drop loss as in Equation (2).

However, before coming to that conclusion, we had believed that an issue was happening with initialization. Specifically, in our setup, we match each neuron to a specific category to compute drop losses – could it be possible that these matches are suboptimal due to how the networks are initialized? Thus to do this, we develop a matching algorithm (Algorithms 1 and 2) to prioritize

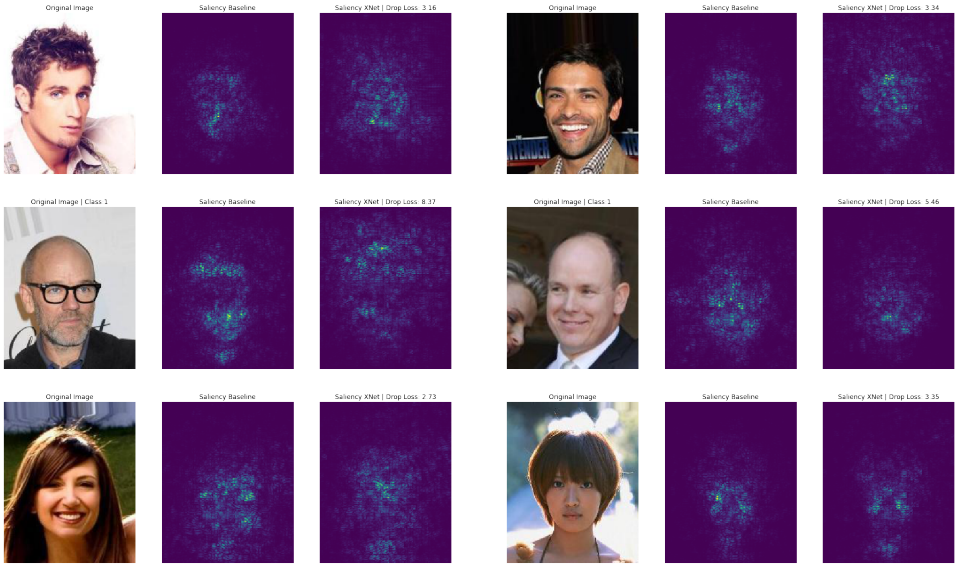

Figure 3: Saliency maps from each category. Rows represent the categories 5 O'Clock Shadow, Bald, and Bangs, respectively. Left group of columns contains promising saliencies from XNet, and the right group of columns contains saliences that do not match up to the desired category for XNet.

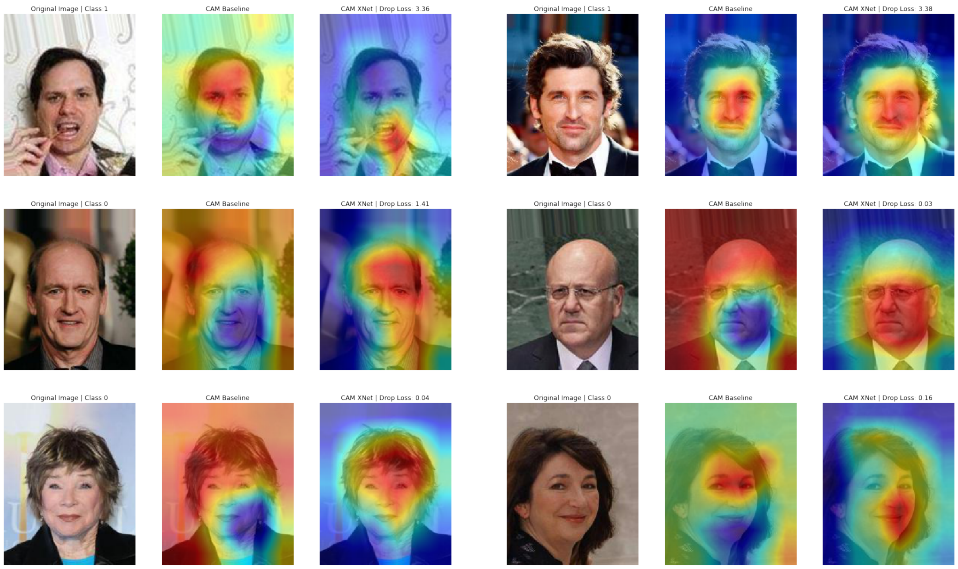

Figure 4: Class activation maps from each category. Format is the same as above, including good and bad examples for XNet in the left and right groups, respectively.

high drop losses. This matching framework can be applied any time before iterating through the training set, thus, we develop two versions: only run it once before all of training (*initial match*), or run matching before every epoch (*EM match*). Note the latter is *EM-like* in that the optimal matches affect the network parameters the next iteration of matching would use.

Now, we recreate Figure 7 utilizing these new algorithms in Figure 8. Despite doing an initial matching state, all it does is setup a different neuron to be optimized (to a higher degree) than the original baseline training – the two other neurons still decrease in drop loss. However, once running the EM-like matching algorithm, we see an interesting phenomena: after certain epochs, the

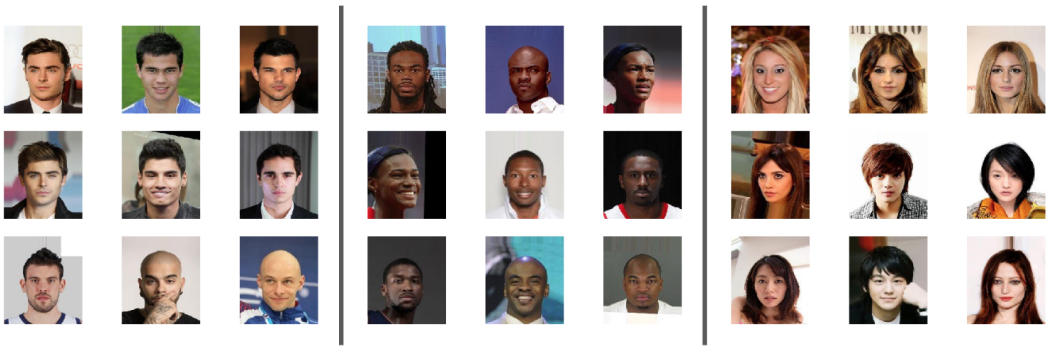

Figure 5: Top 9 images from XNet that induce the highest drop loss for each neuron. From left to right, we expect to see 5 O'Clock Shadow, Bald, and Bangs. The most convincing is bangs.

Figure 6: Same as Figure 5, except from the baseline training.

matching algorithm switches the neuron to optimize, but each time results in a worse drop loss for the two others. The sharp declines after each iteration is probably due to the fact that the categories new neurons are matched to have not yet been captured by XNet; the sharp inclines would have a similar reason, in that the neuron gets lucky and matches to the captured category. This indicates that although a certain neuron was conditioned to have high drop loss on a category, it also made

Algorithm 1 Compute Drop Losses for each Neuron per Category

Require: Network parameters θ, dataset X

function COMPUTE-DROPLOSSES(X, θ)
  $M \leftarrow$ Matrix of size |Categories| x |Neurons|. Initialize with 0s.
  for minibatch $\mathcal{X}$ in X **do**
    for each Category j **do**
      for each Neuron i **do**
        $M[j, i] \mathrel{+}= \mathcal{L}(\hat{\theta}_i, \mathcal{X}_j)$
      end for
    end for
  end for
  for each Category j **do**
    $M[j, :] = \frac{1}{|\text{Category } j \text{ Images}|} M[j, :]$
  end for
  return M
end function

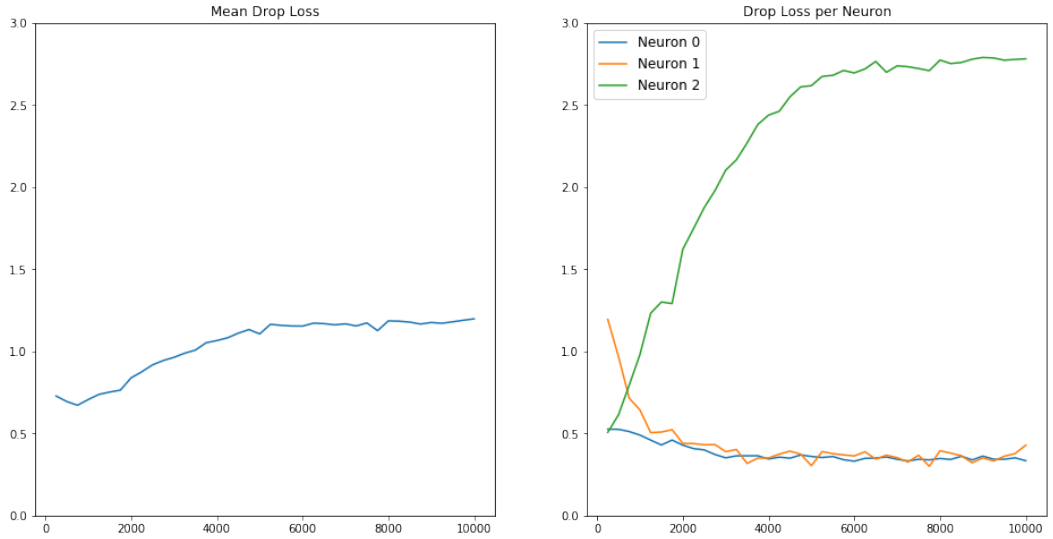

Figure 7: Drop Losses during training with Equation (1). **Left:** The mean drop loss gradually increases over training. **Right:** Only a single neuron is increasing its drop loss.

Algorithm 2 Bipartite Matching for Neurons and Categories

Require: Drop loss matrix M as described in Algorithm 1

1: **function** MATCH(M)

2: Run bipartite matching algorithm, optimizing for minimum sum using $-M$ as weights. **return** optimal matches from Line (2)

3: **end function**

the other neurons sensitive as well. For these reasons, we did not end up using these in the final algorithm, and instead opted with a simple minimum as in Equation (2).

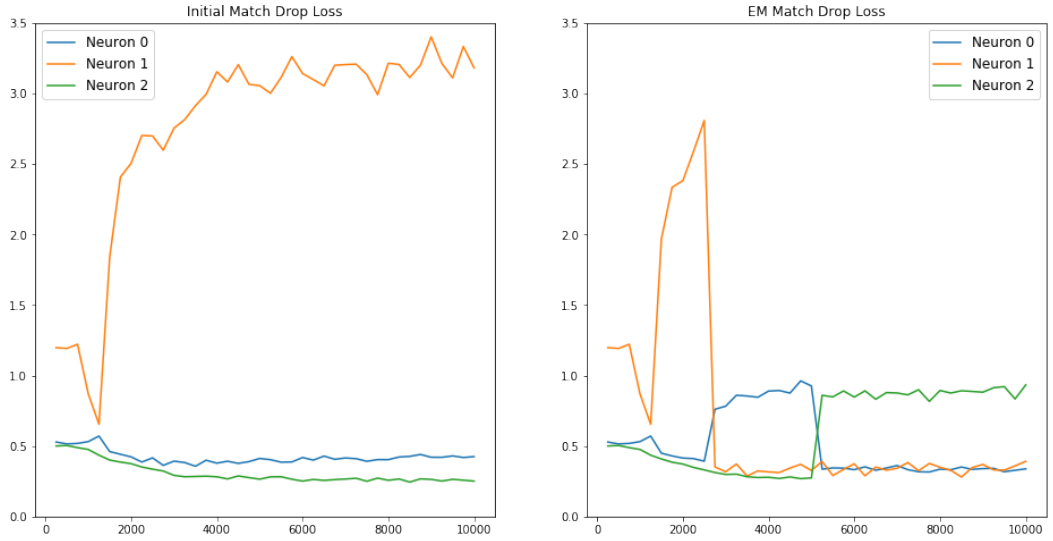

Figure 8: Drop Losses using Matching. **Left:** Initial Matching Losses. **Right:** EM Matching Losses. We observe the same phenomena as in Figure 7, except on the right, we see different neurons being prioritized between epochs.

5 Conclusion and Future Work

We have shown a novel training framework that involves zeroing out internal representations within a neural network. We optimize for a significant increase in classification loss, indicating that the neuron must have captured useful information for classification related to a certain category. We verify this through training dynamics, but also attempt to visualize the interpretability of the network through standard *post-hoc* techniques.

One thing that must be considered in the future is: what are the neurons in XNet actually learning? These neurons are trying to capture *category-specific* features that would be good for classification. These features may not be visually seen in the original image space, and may be based off of correlations found within the category itself. Thus, more work must be done in actually identifying what the neurons focus on. One possibility is to use image space optimization methods such as Olah et al. (2017). i.e. produce images that would most activate certain neurons. In regards to this, our current method does not necessarily make things more interpretable. Another aspect we could work on is actually incorporating something during optimization that would make these methods easier to understand for its human users.

In addition, one big disadvantage to the current method is that it still requires extra category labels to be trained with. In most cases, data collection for the classification task can already be difficult, thus limiting the flexibility of XNet. To allievate this, we should think of ways to make the category specification unsupervised, but this is a much harder problem to tackle and is related to recent work in representation learning. Another avenue to use is to find problems that actually require additional labels/features. One possibility is in fairness – often protected attributes such as race and gender are already collected in the dataset, but companies are now moving towards debiasing their models with respect to these categories. As seen in Figure 5, we could possibly use a similar framework to XNet to *debias* a neural network's internal representations. Regardless, these are the main issues that have come up during this project, and hopefully in the future we can come up with solutions to these problems.

References

Sebastian Bach, Alexander Binder, Grégoire Montavon, Frederick Klauschen, Klaus-Robert Müller, and Wojciech Samek. On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. *PLoS ONE*, 10(7):e0130140, 07 2015. doi: 10.1371/journal.pone.0130140. URL http://dx.doi.org/10.1371%2Fjournal.pone.0130140.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015. URL http://arxiv.org/abs/1512.03385.

Oscar Li, Hao Liu, Chaofan Chen, and Cynthia Rudin. Deep learning for case-based reasoning through prototypes: A neural network that explains its predictions. *CoRR*, abs/1710.04806, 2017. URL http://arxiv.org/abs/1710.04806.

Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *Proceedings of International Conference on Computer Vision (ICCV)*, December 2015.

Chris Olah, Alexander Mordvintsev, and Ludwig Schubert. Feature visualization. *Distill*, 2017. doi: 10.23915/distill.00007. https://distill.pub/2017/feature-visualization.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. In *NIPS-W*, 2017.

Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. *CoRR*, abs/1312.6034, 2013. URL http://arxiv.org/abs/1312.6034.

Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian J. Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *CoRR*, abs/1312.6199, 2014.

Quanshi Zhang, Ying Nian Wu, and Song-Chun Zhu. Interpretable convolutional neural networks. *CoRR*, abs/1710.00935, 2017. URL http://arxiv.org/abs/1710.00935.

B. Zhou, A. Khosla, Lapedriza. A., A. Oliva, and A. Torralba. Learning Deep Features for Discriminative Localization. *CVPR*, 2016.