MEMORIZATION AND GENERALIZATION IN DEEP CNNS USING SOFT GATING MECHANISMS

Prachi Garg
Delhi Technological University, New Delhi, India

Shivang Agarwal
Normandie Univ, UNICAEN, ENSICAEN, CNRS

Alexis Lechervy
Normandie Univ, UNICAEN, ENSICAEN, CNRS

Frederic Jurie
Normandie Univ, UNICAEN, ENSICAEN, CNRS

September, 2019

ABSTRACT

A deep neural network learns patterns to hypothesize a large subset of samples that lie in-distribution and it memorises any out-of-distribution samples. Previous works [1] show that the effective capacity of neural networks is sufficient to completely memorize even random noise in the training data. Memorization of the out-of-distribution samples results in an over-complicated decision boundary which doesn't generalize well on the test set. While fitting to noise, the generalisation error increases and the DNN performs poorly on test set. In this work, we learn the memorizable and generalizable samples differently. While the initial layers that are common to all examples tend to learn general patterns, we dedicate certain deeper additional layers in the network to memorise the out-of-distribution examples. The proposed model uses a soft gating mechanism to decide on the fly if an input will skip the additional layers or pass through them based on its hardness measure. An entropy based metric is used to assign hardness to each example. We hypothesize that this separation of the generalizable and memorizable samples will lead to a simplification of the decision boundary learnt and result in better generalization in the neural network. We aim to combine the strengths of both generalisation and memorisation in a single neural network, in an attempt to blend relevance and diversity in the final representation, while inspecting if this improves the generalization performance of the network.

1 Introduction

An easy example is an in-distribution (ID) example which is in abundance in the dataset and is not an outlier or noise. These are the samples from which the DNN learns patterns. If a dataset contains only these, then once a model is trained, the addition of more parameters to the architecture or training for more epochs has very less affect on further enhancing performance [2]. On the other hand, we define a hard example as an out-of-distribution (OOD) example which is either a noisy sample (the label or the image could be noisy) or an outlier which the DNN memorises as it is. Keeping in mind that even a simple depth two neural network with sufficient number of parameters has finite sample expressivity [1], as the percentage of hard examples increases in the dataset, either the model parameters or the number of training epochs has to be increased to fit the model on the train set.

Even when the dataset contains a reasonable amount of noisy samples, the model is known to learn the general patterns (easy samples) first and then learn to fit the noise. This was shown in the experiments with Critical Sample Ratio in the paper [2]. From the experiments in [2], it can be concluded that large capacity DNNs fit noise examples (representative of hard examples) in a way that is different from and doesn't interfere with the learning of easy examples. It has also been shown that if the dataset has only real (in-distribution) data, it gives a fairly good accuracy even when we use relatively smaller CNNs, with only a few layers to fit the data. Hence, the CNN can be confident about an image after only a few initial layers. In such cases, passing it through the rest of the large network is an overhead which if avoided can lead to better training.

It is also a well established fact that shallow CNN layers extract low level generic features which are common for most samples while the deeper layers extract categorical features which are more input-dependent. Both the above noted observations indicate the need to explore architectures which will handle the ID and OOD examples differently. It is evident that a DNN tries to learn patterns from the easy examples and hence we can say that the ID examples are generalised whereas it tries to dedicate the later layers (or additional parameters as they are added) to memorise the hard or OOD examples. Hence easy examples can be associated with generalisation and hard examples pertain to memorisation in DNNs. This forms the basis of this work where the primary aim is to handle the generalisable and memorisable samples differently. We have a two-fold objective. First, to devise a deep CNN architecture that will handle the ID and OOD examples differently by dedicating certain deeper layers to be used more exclusively by the OOD examples while the ID examples skip them. Second, to understand qualitatively if the network can be made to generalise and memorise differently and whether this will result in a performance increase from the baselines.

We achieve the above stated objectives using soft gating mechanisms on deep Residual networks which allow the CNN to decide dynamically whether each input sample is ID or OOD and correspondingly skip or pass through the deeper network layers. This enables us to incorporate both generalisation and memorisation in a single network and helps us analyse their combined affect on overall network performance.

2 Related Works

The concept of routing information through a DNN on information highways using a gating mechanism for better optimisation and easier training was first introduced in Highway Networks [3]. After the introduction of Deep residual networks [4], gating mechanisms were extensively studied by [5]. Their skip connection experiments in the paper give us a good starting point to understand the kind of gating strategies that have been explored for residual units in the past, their capacity and affect on the overall network performance. These form the fundamentals of soft attention mechanisms in deep residual networks.

Our approach is essentially a conditional computation method for convolutional networks where we study the affect of selectively using or skipping model parameters conditioned on the input. Closely related to our work are architectures [6] and [7] that add classification branches to intermediate layers in the network. BranchyNet exploits the observation that features learned at an early layer of a network may often be sufficient for the classification of a large subset of data population (the ID samples). It augments additional side branch classifiers which allow test samples with high confidence to exit the network early via these branches. Unlike our work, these are adaptive computation time strategies for convolutional networks with the primary aim of improving efficiency of feed forward inference. On the other hand, we focus on investigating architectures that could combine memorisation and generalisation in a single model.

HydraNets [8] demonstrates combining conditional computation with a mixture of parallel experts. It uses multiple category based branches for extracting features and uses a gating module to take decisions on input specific branch selection. ConvNet-AIG [9] is an adaptive computation technique for neural networks which defines its network topology conditioned on the input image. For each input, it uses a binary gating mechanism to dynamically decide if a residual layer will be used to form the inference graph or not. It can be seen as an example of a hard attention mechanism. Our work differs from ConvNet-AIG as the focus here is on qualitatively understanding how a CNN can be made to memorise and generalise differently using soft gating mechanisms.

More recently, GaterNet [10] proposed input dependent dynamic filter selection in deep CNNs. By using binary gates to select filters in the backbone network for processing the input, they force different parts of the model to learn from different types of samples. Another notable inference in GaterNet is the distribution of gates generated at the shallow versus deeper layers. They show that the proportion of input dependent gates increases as the network goes deeper. Our method differs from GaterNet and other such dynamic network configuration techniques as we use soft gating mechanisms rather than explicit binary gates to control which units or layers participate in prediction for an input. While the GaterNet or ConvNet-AIG focus on concepts like different model parameters learning category specific characteristics or reducing computation time, we explore qualitatively how a deep neural network can be made to generalise and memorise in a manner that results in better feature representations and reduces the generalisation error.

Lastly in the paper [11], authors present the Wide & Deep learning framework to achieve both memorization and generalization in one model, by jointly training a linear model component and a neural network component. They emphasise on combining the strengths of both types of models and achieve significant improvement on the recommender system of Google Play.

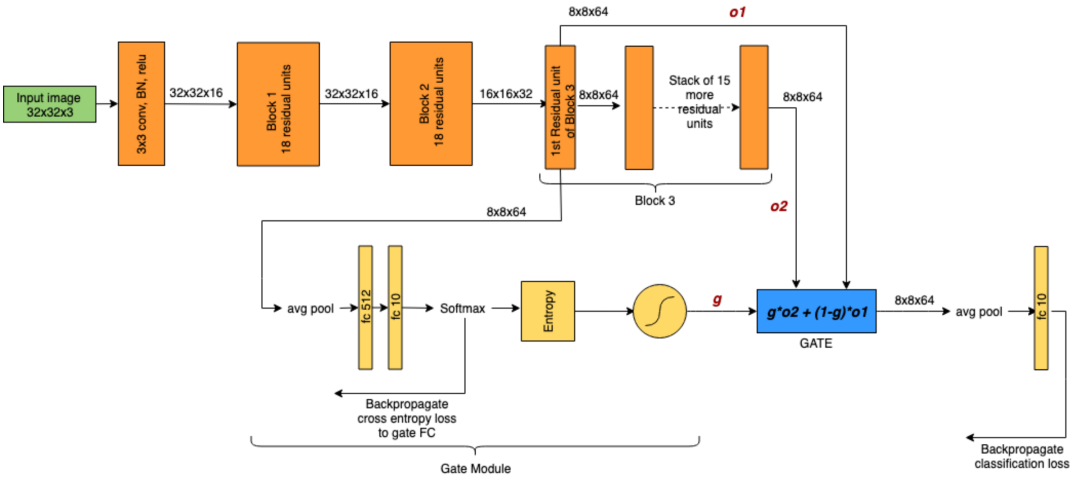

Figure 1: Network architecture for model Containing Entropy Gate with Classifier. This illustration applies the gate on ResNet 110 for Cifar 10

3 The need to handle ID and OOD samples differently

Before we proceed any further, it is important to understand the motivation behind this concept of dedicating different layers of the model for generalisation and separate ones for memorisation. There are several samples in the dataset that can be classified by a simple model with lesser number of parameters. The decision boundary of a model learnt this way is a fairly simple, general one that doesn't overfit and performs well on the test set. When the dataset contains outliers that can't be classified using a basic decision boundary, then the model overfits to the dataset resulting in a complex decision boundary that performs poorly on the test set. The basic motivation for this work was to devise an architecture that will fit the train set without making the decision boundary too complicated. The idea is to keep a simple decision boundary for most of the examples (the in-distribution ones) and learn the out-of-distribution samples separately by dedicating different layers to do the task. This way we are learning specific regions of the feature space as it is (memorisation). The aim is to study the combined affect of both these phenomena. So when we evaluate on test set, the easy examples do not suffer misclassification merely due to an over complicated decision surface. While generalisation can be defined as summarising a large number of sparse observations into a compact representation of knowledge, memorisation is associated with learning the exceptions as they are [11]. A comprehensive inclusion of both these concepts ensures a much needed blend of relevance and diversity in the final representation.

4 Model Containing Entropy Gate with Classifier

When the aim is to construct a model that generalises well to the patterns in the train set and memorises outliers in a way that doesn't diminish the performance of the model on the test set samples, the foremost area of concern is the question of how to determine whether a particular sample qualifies as easy or hard. For this distinction, we explore the usage of entropy as a metric to assign a measure of hardness to the samples. Previously, entropy has been used in [12] and [13] to detect out-of-distribution examples.

The entropy of a classifier is one of the simplest ways of determining how easy or hard a sample is. Consider a deep neural network with a softmax classifier at the end. When the output of the softmax (output dimension is same as number of classes) is used to calculate entropy, the entropy value is lower for an easy sample which has high confidence probability of belonging to a particular class and lower confidence in all other classes. On the other hand, the entropy is higher for a hard sample which has higher confidence probabilities for more than 1 class.

4.1 Model Overview

We now formally define the model shown in Figure 1. In this model, a ResNet with 54 residual units stacked together forms the backbone of the model architecture. Recall that the original ResNet-110 [4], consists of 6n layers of 3x3 convolutions stacked together on feature map sizes 32,16,8, with 2n layers for each feature map size. We will refer to each group of consecutive 2n layers as a block, getting 3 blocks B1, B2, B3 each having 36 convolutional layers or 18

residual units if n=18. Initially, we apply the gating mechanism exclusively on the last block (B3). The output of the gate is given by the equation:

$$Gate\ Output = g(x) * o2 + (1 - g(x)) * o1 \tag{1}$$

where $o1$ is the output from the 1st layer of B3, $o2$ is the output of the block B3 and $g(x)$ is the gate control parameter. Equation (1) is a soft gating mechanism. Inspired from the residual skip connections, output o1 skips B3 and is analogous to an identity skip connection in a traditional ResNet whereas o2 is the output of the block B3. The part of the architecture before B3 represents the smaller network that is common for all samples in the dataset. Since generalisation on the in-distribution samples is done fairly well by networks with a few layers, we can say that the common part of the architecture is dedicated to learning patterns and is representative of a simple general hypothesis that can classify a wide range of in-distribution examples. We construct the gate module in a way that only the relatively hard examples participate in the forward pass through the additional layers represented by the layers in B3 (excluding 1st residual unit which is common for all samples). All other samples skip these B3 layers. Intuitively, the additional part of the architecture should get specialised in memorising noise samples or outliers.

The way the above concept has been implemented is by using a dedicated gating module consisting of fully connected layers with a 10 way (for cifar 10) softmax classifier to compute the gate control parameter, referred to as g(x). At the gate, an in-distribution example will have a low value of g(x) (less than 0.5) due to which the second term in equation (1) dominates the gate output and the gate output favours the affect of o1. This is equivalent to the gate skipping B3. Similarly, when a hard example is passed through the model, it results in a higher g(x) value (greater than 0.5). A higher g(x) value nullifies the affect of the second term in equation (1) and o2 dominates the gate output. Hence, hard examples pass through the layers in B3 giving them additional model parameters for memorisation.

4.2 Loss computation and backpropagation

The gate module is trained independent of the rest of the network using a classifier of its own. It uses target labels to get cross entropy loss and back propagates this loss called the gate loss only to the layers in the gate module. The idea is that as the gate classifier gets better at predicting the true labels i.e. as the cross entropy loss for the gate reduces, the softmax value increases for a particular class and this results in a decrease in entropy. By constructing such an architecture, we aim that most of the samples should skip the gate. The final classification loss at the end of the model is backpropagated through the entire network except the layers in the gate module. We conduct experiments where for each final classification loss backpropagation through the network, the gate loss is backpropagated multiple times. This is done because we suspect that the gate layers might require more training iterations as compared to the rest of the network. More details will be covered in the experiments section.

5 Experiments

We conduct experiments on CIFAR[14] with ResNet [4]. In all experiments, the original ResNets form the baseline on which we add gating modifications and train models. Our models outperform the baselines on CIFAR 10. On CIFAR 100, the results are found to be incomprehensible due to large standard deviation of test accuracy about the mean.

5.1 Datasets

CIFAR 10 and CIFAR 100 consist of natural images that belong to 10 and 100 classes respectively. There are 50,000 training and 10,000 test images in each of these datasets. In CIFAR 10, there are 5,000 training images for each class whereas there are 500 training images per class in CIFAR 100. Each image is a 32 x 32 coloured image and we normalize them using the channel means and standard deviations. Standard data augmentation by random cropping and horizontal flipping are applied to the training set. Only normalization is applied to test and validation set images. We randomly select and fix a set of 5,000 images as validation set. We use the validation set for hyper parameter selection. We report final results in this section after training on the complete training set. All test set performance accuracies reported on ResNet 110 for CIFAR 10 and on ResNet 164 for CIFAR 100, including the standard baselines are averaged over five runs with their respective standard deviation.

5.2 Experiments with gate over B3

Only Block B3 has been modified. The gate FC for B3 has 512 units in the intermediate FC layer.

5.2.1 CIFAR 10

For CIFAR 10, we conduct experiments on ResNet 110 with basic blocks (6n+2) layers. We modify block B3. We experiment with several hyperparameter combinations. As mentioned earlier, we trained models where the gate loss was backpropagated multiple times for each backpropagation of the final classification loss. We use gate_iter:godel_iter ratios of 1, 3 and 5 where a gate_iter:model_iter=5 implies that for each forward and backward pass of classification loss, the gate layers were trained 5 times. We use 2 initialisation schemes, first being Kaiming He initialisation as used in [4]. Second, we initialise the models with the baselines trained on the same respective dataset. Results of the 2 types of initialisation schemes can be seen in table 1. It can be observed that we get the best performance at 94.12% test accuracy when we initialise from scratch. Table 1 has gate_iter:model_iter=1. Performance degrades when we use gate_iter:model_iter=3 and further degrades on using a ratio of 5.

	Gate Initial LR	Gate LR Schedule	Test Accuracy	Standard Deviation
Baseline			93.478	0.2644239021
Scratch Initialisation	0.01	constant	**94.12**	0.189076704
	0.01	reduce, 50 epochs	94.078	0.07155417528
	0.001	constant	94.07	0.09695359715
	0.001	reduce, 50 epochs	93.906	0.3282224855
Baseline Initialisation	0.01	constant	**94.012**	0.2610938529
	0.01	reduce, 50 epochs	94.006	0.09208691547
	0.001	constant	93.942	0.2096902477
	0.001	reduce, 50 epochs	93.94	0.3040559159

Table 1: CIFAR 10 on ResNet 110, Gate over B3

5.2.2 CIFAR 100

For CIFAR 100, first we conducted experiments on ResNet 164 with bottleneck blocks (9n+2) layers. We use gate_iter:model_iter=1. The results of both initialisation schemes are given in Table 2. All results are mean of 5 runs. The models perform close to the baseline but we can't comprehend well as the standard deviations are very high (1.0). We also perform experiments on ResNet 110 as the standard deviation of CIFAR 100 trained on ResNet 110 is lesser. This is shown in Table 3. The best performing hyperparameter combinations perform close to the baseline but fail to surpass the baseline. Hence, CIFAR 100 doesn't show the same response on this architecture as CIFAR 10.

	Gate Initial LR	Gate LR Schedule	Test Accuracy	Standard Deviation
Baseline			74.466	0.9917308103
Scratch Initialisation	0.01	constant	71.304	1.074862782
	0.01	reduce, 50 epochs	**74.274**	0.5218524696
	0.001	constant	68.9	1.43469509
	0.001	reduce, 50 epochs	71.286	1.12351235
Baseline Initialisation	0.01	constant	71.348	1.269318715
	0.01	reduce, 50 epochs	**74.392**	0.314436003
	0.001	constant	70.062	0.9408613075
	0.001	reduce, 50 epochs	72.536	1.135134353

Table 2: CIFAR 100 on ResNet 164, Gate over B3

5.3 Experiments with gates over B2 and B3

Next, we apply the gating mechanism over blocks B2 and B3. The gate FC for B2 has 216 units in the intermediate FC layer. The gate FC for B3 has 512 units in the intermediate FC layer. We initialise models from scratch and perform experiments only on CIFAR 10. These are present in Table 4. We report performance after training for a single run. The models outperform the baseline by a margin of 0.4%.

	Gate Initial LR	Gate LR Schedule	Test Accuracy
Baseline			71.816
Scratch Initialisation	0.01	constant	69.24
	0.01	reduce, 50 epochs	**70.59**
	0.001	constant	65.47
	0.001	reduce, 50 epochs	69.87
Baseline Initialisation	0.01	constant	70.03
	0.01	reduce, 50 epochs	71.48
	0.001	constant	69.16
	0.001	reduce, 50 epochs	**71.74**

Table 3: CIFAR 100 on ResNet 110, Gate over B3

	Gate Initial LR	Gate LR Schedule	Test Accuracy
Baseline			93.478
Scratch Initialisation	0.01	constant	93.45
	0.01	reduce, 50 epochs	**93.87**
	0.001	constant	93.83
	0.001	reduce, 50 epochs	93.75

Table 4: CIFAR 10 on ResNet 110, Gates over B2 and B3

5.4 Implementation Details

Training schedule used to train these is similar to the one prescribed in the original papers [4], [5]. We use a weight decay of 0.0001 and momentum of 0.9, adopt the kaiming weight initialisation and BN with no dropout. These models are trained with a mini-batch size of 128 on a single GPU. For all CIFAR experiments on resnet depth greater than or equal to 110, we warm up the training by using a smaller learning rate of 0.01 at the beginning for 391 iterations and go back to 0.1 after that. We divide it by 10 at 46k and 70k iterations or 120 and 180 epochs, and train for a total of 117k iterations (300 epochs). We experiment with variable initial learning rates for training the gate layers, either keeping it constant throughout training or reducing it by 10 every 50 epochs. When training with a 45k:5k training-validation split ratio, we reduce the model learning rate at 133 and 200 epochs, training for a total of 333 epochs.

5.5 Gate Distribution and Conclusion

In this section, we qualitatively prove the hypothesis that the proposed architecture results in good generalization, and differently learns out-of-distribution samples in the dedicated layers, *by construction*. We plot a histogram showing the evolution of the gate values g for all training samples, over the training process (from first to last training epoch). As can be observed in Figure 2, initially the gate values are higher, i.e. there is a large proportion of higher entropy values g (> 0.5). As the training proceeds, entropy reduces and at the end very few samples have high g values. Hence, after training, all samples can be said to be skipping the extra layers as per our proposed model logic. Effectively, the initial layers are responsible for feature predictions on a majority of the samples which are in-distribution, and performance of these predictions is competitive with respect to the vanilla ResNet baselines. Hence, in this work, we have proposed a novel architecture for better generalization of in-distribution samples and memorization of out-of-distribution samples.

References

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

[2] Devansh Arpit, Stanisław Jastrzębski, Nicolas Ballas, David Krueger, Emmanuel Bengio, Maxinder S Kanwal, Tegan Maharaj, Asja Fischer, Aaron Courville, Yoshua Bengio, et al. A closer look at memorization in deep networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 233–242. JMLR. org, 2017.

[3] Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. Highway networks. *arXiv preprint arXiv:1505.00387*, 2015.

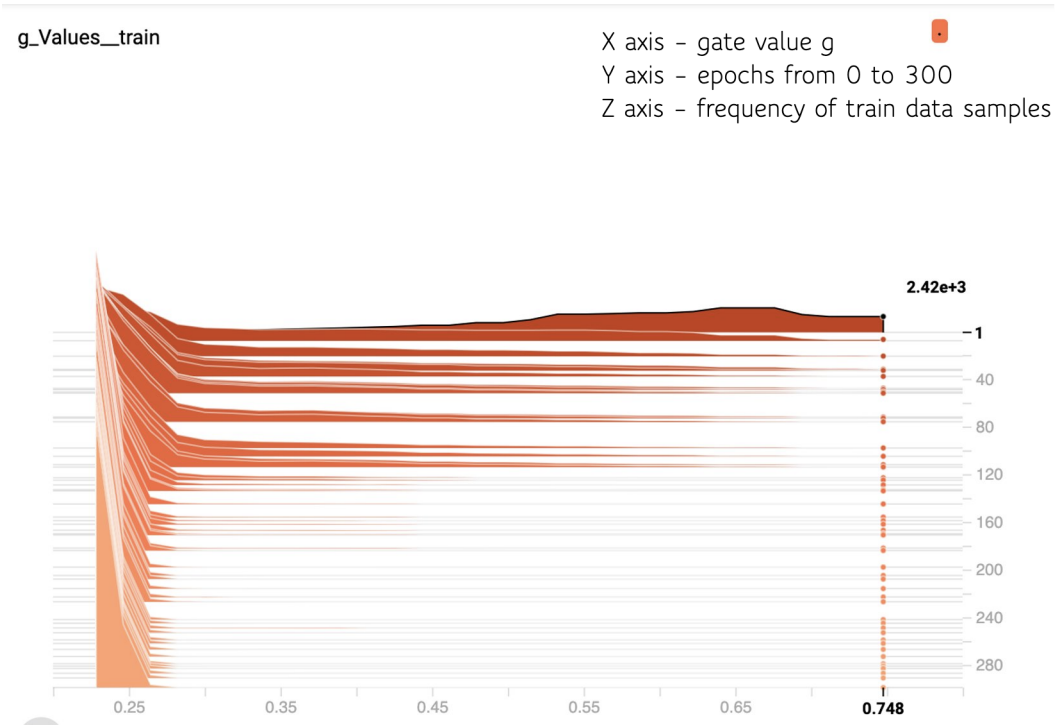

Figure 2: Histogram showing distribution of gate values g as training proceeds over 300 epochs (plotted for train set)

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

[5] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. In *European conference on computer vision*, pages 630–645. Springer, 2016.

[6] Gao Huang, Danlu Chen, Tianhong Li, Felix Wu, Laurens van der Maaten, and Kilian Q Weinberger. Multi-scale dense networks for resource efficient image classification. *arXiv preprint arXiv:1703.09844*, 2017.

[7] Surat Teerapittayanon, Bradley McDanel, and Hsiang-Tsung Kung. Branchynet: Fast inference via early exiting from deep neural networks. In *2016 23rd International Conference on Pattern Recognition (ICPR)*, pages 2464–2469. IEEE, 2016.

[8] Ravi Teja Mullapudi, William R Mark, Noam Shazeer, and Kayvon Fatahalian. Hydranets: Specialized dynamic architectures for efficient inference. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8080–8089, 2018.

[9] Andreas Veit and Serge Belongie. Convolutional networks with adaptive inference graphs. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 3–18, 2018.

[10] Zhourong Chen, Yang Li, Samy Bengio, and Si Si. You look twice: Gaternet for dynamic filter selection in cnns. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9172–9180, 2019.

[11] Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hrishi Aradhye, Glen Anderson, Greg Corrado, Wei Chai, Mustafa Ispir, et al. Wide & deep learning for recommender systems. In *Proceedings of the 1st workshop on deep learning for recommender systems*, pages 7–10. ACM, 2016.

[12] Wenhu Chen, Yilin Shen, William Wang, and Hongxia Jin. A variational dirichlet framework for out-of-distribution detection. 2018.

[13] Apoorv Vyas, Nataraj Jammalamadaka, Xia Zhu, Dipankar Das, Bharat Kaul, and Theodore L Willke. Out-of-distribution detection using an ensemble of self supervised leave-out classifiers. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 550–564, 2018.

[14] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009.