INFORMATION DISTANCE FOR NEURAL NETWORK FUNCTIONS

Anonymous authors
Paper under double-blind review

ABSTRACT

We provide a practical distance measure in the space of functions parameterized by neural networks. It is based on classic information distance and we replace the uncomputable Kolmogorov complexity in the original definition with information measured by codelength of prequential coding. Empirically, we show that information distance is invariant with respect to different parameterization of the functions with neural networks. We also verify that information distance can faithfully reflect similarities of neural network functions. Finally, we applied information distance to investigate the relationship between neural network models, and demonstrate the connection between information distance and multiple characteristics and behaviors of neural network models.

1 INTRODUCTION

Deep neural networks can be trained to represent complex functions that describe sophisticated input-output relationships, such as image classification and machine translation. Because the functions are highly non-linear and are parameterized in high-dimensional spaces, there is relatively little understanding of the functions represented by deep neural networks. One could interpret deep models by linear approximations (Ribeiro et al., 2016), or from the perspective of piece-wise linear functions, such as in (Arora et al., 2018).

If the space of functions representable by neural networks admits a distance measure, then it would be a useful tool to help analyze and gain insight about neural networks. A major difficulty is the vast number of possibilities of parameterizing a function, which makes it difficult to characterize the similarity given two networks. Measuring similarity in the parameter space is straightforward but is restricted to networks with the same structure. Measuring similarity at the output is also restricted to networks trained on the same task. Similarity of representations produced by intermediate layers of networks is proved to be more reliable and consistent (Kornblith et al., 2019), but is not invariant to linear transformations and can fail in some situations, as shown in our experiments.

In this paper, we provide a distance measure of functions based on information distance (Bennett et al., 1998), which is independent of the parameterization of the neural network. This also removes the arbitrariness of choosing "where" to measure the similarity in a neural network. Information distance has mostly been used in data mining (Cilibrasi & Vitányi, 2007; Zhang et al., 2007). Intuitively, information distance measures how much information is needed to transform one function to the other. We rely on prequential coding to estimate this quantity. Prequential coding is a method for efficiently encoding neural networks and datasets. If we regard prequential coding as a compression algorithm for neural networks, then the codelength can give an upper bound of the information quantity in a model.

We propose a method for calculating an approximated version of information distance with prequential coding for arbitrary networks. Then we perform experiments to demonstrate that this information distance is invariant to the parameterization of the network while also being faithful to the intrinsic similarity of models. Using information distance, we are able to sketch a rough view into the space of deep neural networks and uncover the relationship between datasets and models. We also found that information distance can help us understand regularization techniques, measure the diversity of models, and predict a model's ability to generalize.

2 METHODOLOGY

Information distance measures the difference between two objects by information quantity. The information distance between two functions f_A and f_B can be defined as (Bennett et al., 1998):

$$d(f_A, f_B) = max\{K(f_A|f_B), K(f_B|f_A)\} \tag{1}$$

This definition makes use of Kolmogorov complexity: $K(f_B|f_A)$ is the length of the shortest program that transforms f_A into f_B, and information distance d is the larger length of either direction. (Note that this is not the only way to define information distance with Kolmogorov complexity, however we settle with this definition for its simplicity.) Intuitively, this is the minimum number of bits we need to encode f_B with the help of f_A, or how much information is needed to know f_B if f_A is already known.

Given two functions $f_A : X \to Y$ and $f_B : X \to Y'$ defined on the same input space X, each parameterized by a neural network with weights θ_A and θ_B, we want to estimate the information distance between f_A and f_B. The estimation of Kolmogorov complexity term is done by calculating the codelength of prequential coding, so what we get is an upper bound of d, which we denote by d_p (p for prequential coding).

Now we offer a method for estimating an upper bound of $K(f_B|f_A)$ with prequential coding. To send f_B to someone who already has f_A, we generate predictions y_i from f_B using input x_i sampled form X. Assume $\{x_i\}$ is known, we can use prequential coding to send labels $\{y_i\}$. If we send enough labels, the receiver can use $\{x_i, y_i\}$ to train a model to recover f_B.

If it turns out that f_A and f_B have something in common, i.e. $K(f_B|f_A) < K(f_B)$, then with the help of f_A we can reduce the codelength used to transmit f_B. A convenient way of doing so is to use θ_A as the initial model in prequential coding. Now the codelength of k samples is:

$$L^{\text{preq}}(y_{1:k}|x_{1:k}) := -\sum_{i=1}^{k} \log p_{\hat{\theta}_i}(y_i|x_{1:i}, y_{1:i-1}) \tag{2}$$

where $\hat{\theta}_i$ is the parameter of the model learned on data samples from 1 to i, and we set $\hat{\theta}_0 = \theta_A$. With sufficient large k, the function parameterized by $\hat{\theta}_k$ would converge to f_B.

If both θ_A and θ_B are classification networks, we can sample y from the probabilistic distribution of the softmax output of θ_B. In this case, the above codelength not only transmits f_B, but also k specific samples we draw from the output of θ_B. The information contained in the specific samples is $\sum_{i=1}^{k} \log p_{\theta_B}(y_i|x_i)$. Because we only care about estimating $K(f_B|f_A)$, using the "bits-back protocol"(Hinton & van Camp, 1993) the information of samples can be subtracted from the codelength, resulting in an estimation of:

$$L_k(f_B|f_A) = -\sum_{i=1}^{k} \log p_{\hat{\theta}_i}(y_i|x_{1:i}, y_{1:i-1}) + \sum_{i=1}^{k} \log p_{\theta_B}(y_i|x_i) \tag{3}$$

The expectation of codelength L_k over all possible samples from f_B is:

$$\mathbb{E}_{y_{1:k}\sim f_B(x_{1:k})}[L_k(f_B|f_A)] = -\sum_{i=1}^{k} \mathbb{E}_{y_i\sim f_B(x_i)} \log p_{\hat{\theta}_i}(y_i|x_{1:i}, y_{1:i-1}) \tag{4}$$

$$+ \sum_{i=1}^{k} \mathbb{E}_{y_i\sim f_B(x_i)} \log p_{\theta_B}(y_i|x_i) \tag{5}$$

$$= \sum_{i=1}^{k} D_{KL}(f_B(x_i)||f_{\hat{\theta}_i}(x_i)) \tag{6}$$

The expected codelength can be written in forms of the KL-divergence between the output distribution of f_B and $f_{\hat{\theta}_i}$. Another interpretation of the above estimation is, we finetune model θ_A with an increasing number of labels generated by θ_B, and aggregate the KL-divergence between the two models along the way. The more information f_A shares with f_B, the faster the KL-divergence goes to zero, resulting in a lower estimation of $K(f_B|f_A)$.

Here we see a connection between L and the *information transfer* measure L_{IT} (Zhang et al., 2020):

$$L_{IT}^{k}(\theta_n) = L_{\theta_0}^{\text{preq}}(y_{1:k}|x_{1:k}) - L_{\theta_n}^{\text{preq}}(y_{1:k}|x_{1:k}) \tag{7}$$

as $n \to \infty$, $\theta_n \to \theta_B$, and when $y_i \sim f_B(x_i)$, we have

$$\mathbb{E}[L_{IT}^{k}(\theta_n)] = \mathbb{E}[L_{\theta_A}^{\text{preq}}(y_{1:k}|x_{1:k})] - \mathbb{E}[L_{\theta_B}^{\text{preq}}(y_{1:k}|x_{1:k})] \tag{8}$$

$$= \mathbb{E}[L_k(f_B|f_A)] \tag{9}$$

In practice, we can use k sufficiently large such that $f_{\hat{\theta}_k}$ becomes similar to f_B within a threshold:

$$\mathbb{E}_x[D_{KL}(f_B(x)||f_{\hat{\theta}_k}(x))] \leq \epsilon \tag{10}$$

Now we have an approximation of the information distance d using the expected codelength:

$$d_p(f_A, f_B) \triangleq \max\{\mathbb{E}[L_k(f_A|f_B)], \mathbb{E}[L_k(f_B|f_A)]\} \tag{11}$$

2.1 Properties of d_p

The original information distance d, applied to functions defines a metric on the space of functions. Now we check if d_p satisfy the axioms of a metric:

1. $d_p(f_A, f_B) = 0 \Leftrightarrow f_A = f_B$: $f_A = f_B$ if and only if they always produce the same predictions, which is equivalent to $d_p(f_A, f_B) = 0$
2. $d_p(f_A, f_B) = d_p(f_B, f_A)$: by definition.
3. $d_p(f_A, f_B) \leq d_p(f_A, f_C) + d_p(f_C, f_B)$: whether d_p keeps this property of d depends on the efficiency of prequential coding, which in turn depends on model optimization.

Another important property of the information distance d is the invariancy with respect to the parameterization of function f. We found that d_p is also largely invariant to parameterization of the functions. d_p can be used to compare models trained differently, having different structures, or even trained on different tasks. The only condition is that both models should have sufficient expressibility to allow approximation of each other.

2.2 Data-dependency and equivalent Interpretations of data

In machine learning, we often only care about the output of a model on the data distribution of the task. Neural network models are trained on input data from a specific domain, for example, image classification models take natural images in RGB format as valid input. It would be meaningless to discuss the behavior of such a model on non-RGB images. This is an important distinction between a neural network function and a function in the mathematical sense.

This motivates us to take a data-dependent formulation of distance measure. In this paper, we limit our discussion to distribution-dependent information distance:

$$d_p(f_A, f_B) = \max\{K(f'_A|f_B), K(f'_B|f_A)\} \tag{12}$$

where

$$f'_A = \arg\min_{f \in \mathcal{F}_A} K(f|f_B), \qquad f'_B = \arg\min_{f \in \mathcal{F}_B} K(f|f_A) \tag{13}$$

are equivalencies of f_A and f_B in the below function family:

$$\mathcal{F}_\square = \{f | \mathbb{E}_{x \sim D}[D_{KL}(f(x)||f_\square(x))] \leq \epsilon\} \tag{14}$$

$\mathcal{F}_\square$ is a set containing all the functions producing outputs almost indistinguishable from $f_\square$, in the expected sense over x drawn from data distribution D. Because they produce almost identical outputs for $x \sim D$, we call them equivalent interpretations of data in D. Intuitively, this means that instead of transmitting f_B, we can transmit f'_B, which is equivalent to f_B on D, if f'_B can be transmitted in fewer bits.

A quick note on why data-dependency here in the context of neural network models does not break the definition of information distance: if f is a neural network trained on dataset D, then f is fully determined by the information in D plus a random seed (which is of negligible information).

By introducing data-dependency, it enables us to approximate Kolmogorov complexity by coding samples drawn from data distribution D that is used to train the models.

3 EMPIRICAL STUDY

The proposed information distance d_p relies on estimating an upper-bound of Kolmogorov complexity with prequential codelength, therefore we validate the performance of d_p mainly with experiments. We show the advantages of d_p and in what situations are d_p useful.

3.1 ACHIEVING PRACTICAL CODELENGTHS

To make the upper-bound useful, we try to achieve lower codelength in prequential coding by optimizing the model estimation process in sequential coding. We performed a hyper-parameter search to select optimal hyper-parameters that result in the lowest codelength. When the number of samples we have from D is small, there may not be enough examples to train f_A to converge to f_B. We overcome this by using multiple samples per input and training using KL-divergence as objective. We leave detailed discussion of the techniques used in Appendix A. These techniques also allow us to use a lower k to save computation time. Unless otherwise stated, we use $k = 10000$ in experiments throughout this paper.

3.2 INVARIANCES OF INFORMATION DISTANCE

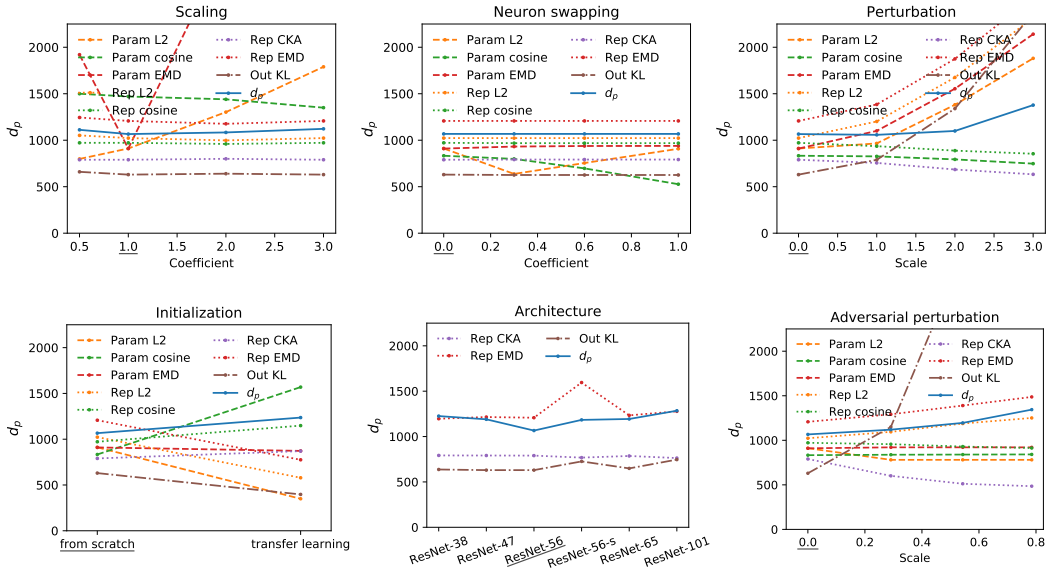

Figure 1: Distance d_p changes with respect to changes in the parameterization of the networks. We measure the distance between a pair of ResNet-56 models θ_A and θ_B, trained on two subsets of Tiny-ImageNet, respectively. We modify the configuration of θ_A while keeping θ_B the same. Modifications include: **scaling**, where we multiply the weights of the network by a coefficient c, **neuron swapping**, where we randomly permute a fraction c of the units within a layer, **perturbation**, where Gaussian random noise of zero mean and standard deviation c is added to the weights of the network, and **adversarial perturbation**, where a vector of standard deviation c is added to the weights of the network to maximize the change of second-to-last layer representations. We also include training θ_A with different **initializations** and using different network **architectures**. Other distance measures used as baselines are: **parameter space distances**, including L2 distance, cosine distance, and Earth Mover Distance (Monge, 1781; Rubner et al., 1998) between parameter vectors, **representation space distances**, which also include L2, cosine, EMD and Linear CKA (Kornblith et al., 2019) on the second-to-last layer representations, and **output space distances** which is the KL-divergence between softmax probabilities. Underlined labels on the x-axis denote the distances measured between unmodified θ_A and θ_B. Curves for each measure are individually scaled (only scaled, not shifted) to ease viewing on the same graph. Cosine similarity and CKA value are within [0,1] and is inversely correlated with distance.

A prominent advantage of information distance is independent of parameterization. Neural networks like multi-layer perceptrons can have a very large number of different configurations (units, weights) that correspond to the same function. Because there is no "canonical" way of parameterizing neural networks, comparing the input-output functions represented by different neural networks can be difficult by merely looking at the network parameters.

There exist many metrics for measuring the similarity or distance between neural networks. But because neural networks are so versatile, similar neural networks can look similar under some metric while very dissimilar by other metrics. There lacks a universal definition of similarity, which is precisely the problem we try to solve with d_p.

To validate the invariance of d_p, we manually modify the configuration of a neural network and see if distances measured by d_p changes. We also include a number of distance metrics as baselines for comparison. Detailed descriptions and results are shown in Figure 1.

Results in Figure 1 indicates that d_p is relatively stable under different kinds of re-parameterization of the network. Other distance measures all exhibit strong dependency on certain kind of parameterization or is inapplicable for some parameterizations. For re-parameterizations that does not change the function f (scaling, neuron swapping), d_p is also basically unchanged. For adding perturbations to the network, information distance only starts to increases when the perturbation is large enough. This is because only large noise will start to "wipe out" information in the network. d_p is also robust to small adversarial perturbations, while also showing that adversarial perturbations destroy information in the network faster than random noise.

If θ_A is initialized with a pre-trained network on another dataset (here on CIFAR-10 (Krizhevsky & Hinton, 2009)), d_p correctly identifies the distance to increase a little bit, because θ_A carries over some information from CIFAR-10, making θ_A and θ_B slightly less similar. If we use architectures other than ResNet-56 (He et al., 2016) to train θ_A, d_p is largely unchanged. The matching model (ResNet-56) achieves the lowest distance, and the more different the model is (in the number of layers) form ResNet-56, the distance is also higher. This indicates that d_p is largely invariant to model parameterizations while being consistent with intuitive similarities of model families.

3.3 Is information distance faithful?

Having seen that d_p has desirable invariance properties, next we check if d_p is aligned with intuitive indicators of model distance. We experiment with two settings: model interpolation and examining the training progress of models. These can serve as a sanity-check that d_p does reflect changes in the information of models.

As a straightforward way of changing model distances, we choose two ResNet-56 models θ_A and θ_B and perform linear interpolation in parameter space: $\theta = (1-c)\,\theta_A + c\,\theta_B$. As we change c from 0 to 1, the model θ changes smoothly from θ_A to θ_B. We apply d_p to measure the distance of the interpolated model to θ_B along the way. The result is shown in Figure 2.

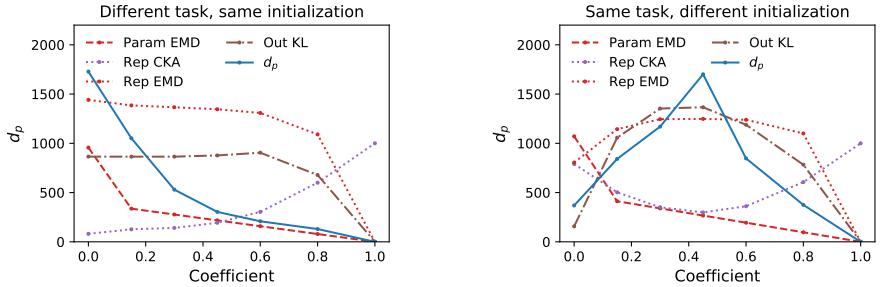

Figure 2: Distance from the interpolation model θ to θ_B. Left: θ_A and θ_B are trained on different tasks, from the same initialization. Right: θ_A and θ_B are trained on the same task but from different initializations.

As we interpolate two functions f_A and f_B in the parameter space, if θ_A and θ_B are parameterized similarly, we observe $d_p(f, f_B)$ to monotonically decrease as f gets closer to f_B (Figure 2 left). At

the beginning when c is small, increase in c introduces more "fresh information" about f_B, thus d_p decreases faster than later in interpolation. On the other hand, if we interpolate two functions that are parameterized differently, because linear mixing of θ_A and θ_B in parameter space leads to a degraded network, the distance would first increase then decrease, indicating a loss of information middle in the interpolation. Overall, the general trend of d_p agrees with advanced similarity measures like EMD and CKA in this scenario.

In the next experiment, we try to examine the training progress of a model. Starting from a random initialization, we train a model n epochs to converge and measure the distance from the model θ_i in i-th epoch to the final model θ_n, as well as from θ_i to the initial model θ_0. The results are shown in Figure 3.

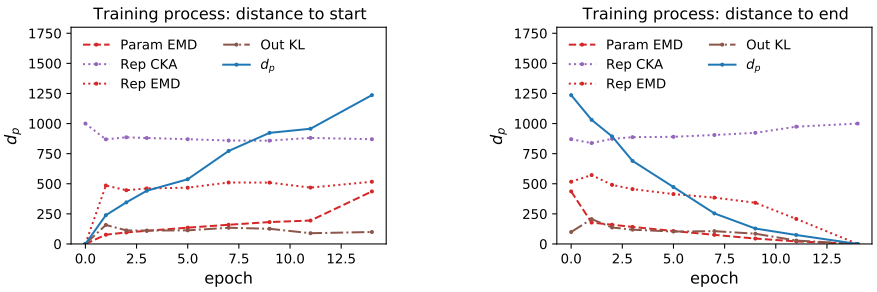

Figure 3: Distance from the i-th epoch model θ_i to initial model θ_0 and final model θ_{14}

As training progresses from the first epoch to the last, $d_p(f_i, f_B)$ steadily decreases and $d_p(f_i, f_A)$ increases. Change in distance is faster in earlier epochs because most of the learning happens in the first few epochs. In this scenario, only d_p is monotonic with training and correctly depicts the training progress. Both EMD and CKA fail to show the dynamics of training beyond the first epoch.

4 Application

To illustrate the utility of a universal function distance, we provide a few scenarios where we use d_p for understanding and making predictions.

4.1 Sketching the geometry of data and model space

A distance measure can help us understand the relationship between datasets and between models. Datasets and models usually live in very high-dimensional spaces, which makes it hard to directly perform a comparison. Instead, we can use d_p to get the information distance between datasets and models. In computer vision there is a myriad of datasets and model structures, and we use the Visual Task Adaptation Benchmark (VTAB) (Zhai et al., 2019) as a collection of vision datasets. On each dataset, a model is trained to represent the input-output function of the task. Then we use d_p to measure pairwise distances between these functions. To help to visualize the relationship, we use Isometric Mapping (Tenenbaum et al., 2000), a manifold learning algorithm to generate three-dimensional embeddings for each function. The distance of points in three-dimensional space is optimized to keep the original structure.

Distances can tell a lot about the relationship between models. In the nine large datasets of VTAB, datasets cluster largely according to the three categories proposed in VTAB (natural, specialized, and structured). CIFAR-100 is very different from any other datasets, but is relatively closer to satellite image datasets than to artificial shape datasets. SVHN (Netzer et al., 2011) is close to 2D shape datasets. The four small datasets are evenly distributed in space: no pair of them is very similar. In terms of model architecture, ResNet variants are relatively similar, while AlexNet (Krizhevsky, 2014) and VGG (Simonyan & Zisserman, 2014) are farther away. VGG with batch normalization is closer to ResNet than without. ResNet-50, ResNeXt-50 (Xie et al., 2017) and WideResNet-50 (Zagoruyko & Komodakis, 2016) are closest as they are indeed very similar.

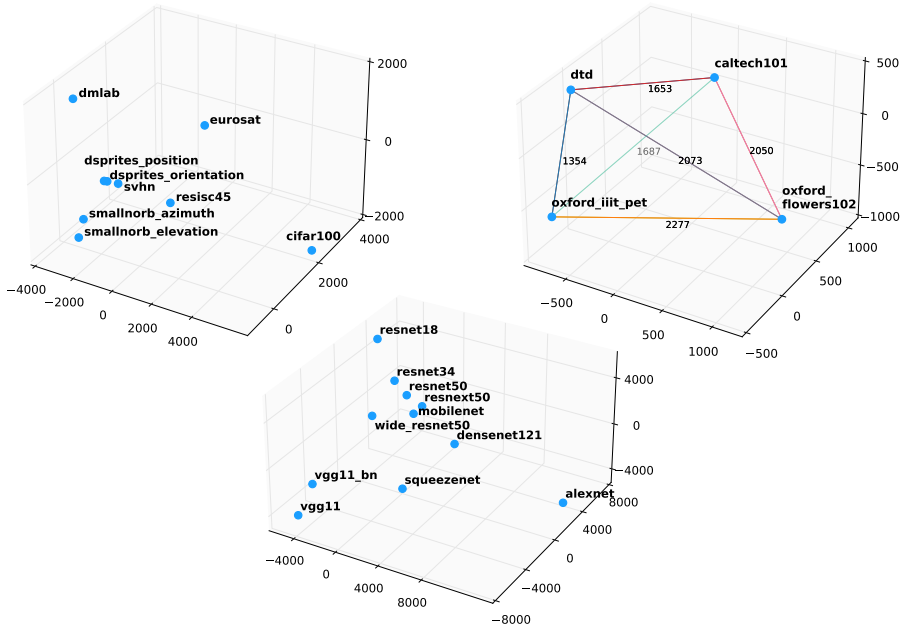

Figure 4: Visualizing distances between datasets and models in three-dimensional space. Top-left: large datasets in VTAB. Top-right: small datasets in VTAB. Bottom: various model architectures trained on ImageNet. The numbers on colored lines are pairwise distances.

4.2 Understanding regularizations

Regularization techniques like L2 regularization can bias the learned neural networks toward less complex functions, while for techniques like dropout (Srivastava et al., 2014) and self-distillation (Furlanello et al., 2018), the regularization effect may be less straightforward to explain. We can use d_p to examine the (information) complexity of a network f, by measuring its distance $d_p(f, \mathbf{0})$ to an empty function.

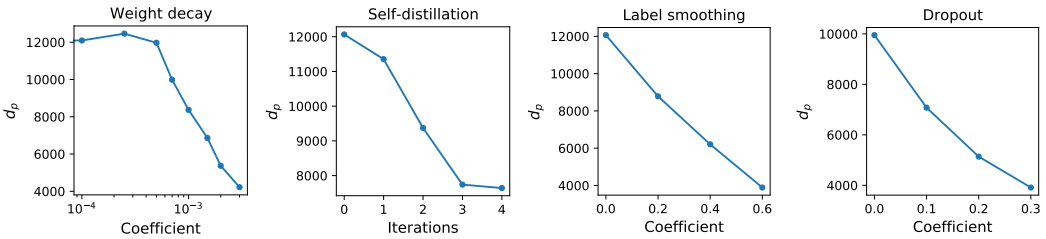

Figure 5: Distance to an empty function $d_p(f, \mathbf{0})$ for models with different kinds of regularization and varying strength. From left to right: Weight decay (L2 regularization), self-distillation, label smoothing, dropout. For dropout, a different base model without batch normalization is used.

From Figure 5, we observe that all the listed techniques result in a reduction of $d_p(f, \mathbf{0})$, which means that the information complexity of the model function f is reduced. For weight decay, information complexity only starts to decrease after the regularization coefficient is larger than a threshold. Self-distillation has a similar effect to regularization, with the number of distillation iterations controlling regularization strength. This agrees with the theoretical analysis in (Mobahi et al., 2020). Label smoothing and dropout also result in simpler models, highlighting their regularization effect.

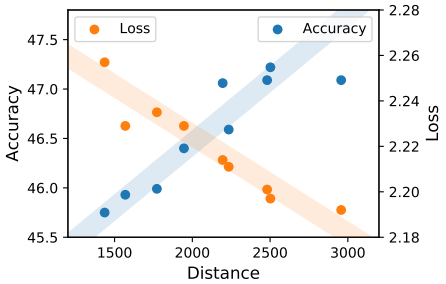

Figure 6: Relation between ensemble performance and model diversity given by $d_p(f_1, f_2)$.

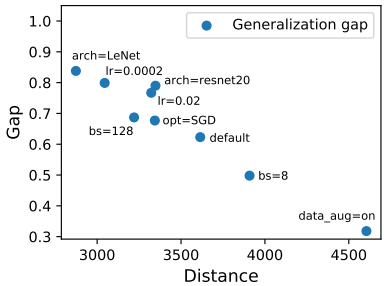

Figure 7: Relation between generalization gap and model complexity $d_p(f, \mathbf{0})$.

4.3 Ensembles and model diversity

Distance can be used as an indicator of model diversity: the larger d_p between models, the more diverse the models are. Ensembling is a common technique to use the consensus of multiple models to deliver superior performance than a single model. We speculate that a larger model diversity will result in more performance gain from ensembles.

To verify this connection, we train a number of models on Tiny-Imagenet, all to the same performance on the validation set, but with different initializations and different subsets of the training set. Then we choose models in pairs to measure their ensemble performance as well as the distance $d_p(f_1, f_2)$ between them. The result is given in Figure 6, and we found a clear correlation between d_p and ensemble performance, and the relationship is about linear. This also indicates that d_p captures model diversity.

4.4 Predicting generalization

Finally, d_p is also linked with model generalization. Generalization of neural networks is heavily affected by hyper-parameters and optimization. There have been several works aiming to find the relationship between generalization performance and properties of the network, but it turns out that predicting generalization gap can be a challenging task (Jiang et al., 2019; 2020).

We perform a small-scale experiment to illustrate the connection between information distance and generalization gap. We train a number of models with different hyper-parameters (batch size, learning rate, optimizer, etc.) all to the same loss on the training set, and then measure the distance to a random model by $d_p(f, \mathbf{0})$. In Figure 7, we observe that the information complexity of the model is also linked with generalization gap, which also turns out to be a roughly linear relationship. Models that generalize better are farther away from a random model than less performing models.

5 Discussion and conclusion

Information distance d_p is based on information distance defined with Komolgorov complexity K. We do not attempt to give a good estimation of K, but instead relying on the efficiency of prequential coding, we empirically illustrate that d_p share the invariance properties of information distance, and reflects the similarity relationships of functions parameterized by neural networks. We also found that d_p is linked with behaviors of models, making it a potential tool for analyzing neural networks.

The most notable difference between d_p and other similarity metrics is universality. Theoretically rooted in information distance, d_p is independent of parameterization and widely applicable in situations involving different tasks and models. However, d_p's utilization of prequential coding also introduces limitations that it might not work in situations where prequential coding fails, for example, when f cannot be efficiently approximated by neural networks.

d_p could introduce a potential scale-free, or even parameterization-free geometry of space spanned by neural models. Optimization with manifold descent by d_p could also remove the dependency on parameterization, thus avoiding ill-posed conditions in some parameterizations (Dinh et al., 2017).

REFERENCES

Raman Arora, Amitabh Basu, Poorya Mianjy, and Anirbit Mukherjee. Understanding deep neural networks with rectified linear units. In *6th International Conference on Learning Representations, ICLR 2018*, 2018.

Charles Beattie, Joel Z. Leibo, Denis Teplyashin, Tom Ward, Marcus Wainwright, Heinrich Küttler, Andrew Lefrancq, Simon Green, Víctor Valdés, Amir Sadik, Julian Schrittwieser, Keith Anderson, Sarah York, Max Cant, Adam Cain, Adrian Bolton, Stephen Gaffney, Helen King, Demis Hassabis, Shane Legg, and Stig Petersen. Deepmind lab. *CoRR*, abs/1612.03801, 2016.

C. H. Bennett, P. Gacs, Ming Li, P. M. B. Vitanyi, and W. H. Zurek. Information distance. *IEEE Transactions on Information Theory*, 44(4):1407–1423, 1998.

Gong Cheng, Junwei Han, and Xiaoqiang Lu. Remote sensing image scene classification: Benchmark and state of the art. *Proceedings of the IEEE*, 105(10):1865–1883, 2017.

Rudi Cilibrasi and Paul M. B. Vitányi. The google similarity distance. *IEEE Trans. Knowl. Data Eng.*, 19(3):370–383, 2007.

Mircea Cimpoi, Subhransu Maji, Iasonas Kokkinos, Sammy Mohamed, and Andrea Vedaldi. Describing textures in the wild. In *2014 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2014*, pp. 3606–3613, 2014.

Laurent Dinh, Razvan Pascanu, Samy Bengio, and Yoshua Bengio. Sharp minima can generalize for deep nets. In Doina Precup and Yee Whye Teh (eds.), *Proceedings of the 34th International Conference on Machine Learning, ICML 2017*, volume 70 of *Proceedings of Machine Learning Research*, pp. 1019–1028, 2017.

Tommaso Furlanello, Zachary Chase Lipton, Michael Tschannen, Laurent Itti, and Anima Anandkumar. Born-again neural networks. In Jennifer G. Dy and Andreas Krause (eds.), *Proceedings of the 35th International Conference on Machine Learning, ICML 2018*, volume 80 of *Proceedings of Machine Learning Research*, pp. 1602–1611, 2018.

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*, pp. 770–778, 2016.

Patrick Helber, Benjamin Bischke, Andreas Dengel, and Damian Borth. Eurosat: A novel dataset and deep learning benchmark for land use and land cover classification. *IEEE J. Sel. Top. Appl. Earth Obs. Remote. Sens.*, 12(7):2217–2226, 2019.

Geoffrey E. Hinton and Drew van Camp. Keeping the neural networks simple by minimizing the description length of the weights. In Lenny Pitt (ed.), *Proceedings of the Sixth Annual ACM Conference on Computational Learning Theory, COLT 1993*, pp. 5–13, 1993.

Geoffrey E. Hinton, Oriol Vinyals, and Jeffrey Dean. Distilling the knowledge in a neural network. *NIPS Deep Learning and Representation Learning Workshop*, 2015.

Gao Huang, Zhuang Liu, Laurens van der Maaten, and Kilian Q. Weinberger. Densely connected convolutional networks. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, pp. 2261–2269, 2017.

Forrest N Iandola, Song Han, Matthew W Moskewicz, Khalid Ashraf, William J Dally, and Kurt Keutzer. Squeezenet: Alexnet-level accuracy with 50x fewer parameters and $<$ 0.5 mb model size. *arXiv preprint arXiv:1602.07360*, 2016.

Yiding Jiang, Dilip Krishnan, Hossein Mobahi, and Samy Bengio. Predicting the generalization gap in deep networks with margin distributions. In *7th International Conference on Learning Representations, ICLR 2019*, 2019.

Yiding Jiang, Behnam Neyshabur, Hossein Mobahi, Dilip Krishnan, and Samy Bengio. Fantastic generalization measures and where to find them. In *8th International Conference on Learning Representations, ICLR 2020*, 2020.

Simon Kornblith, Mohammad Norouzi, Honglak Lee, and Geoffrey E. Hinton. Similarity of neural network representations revisited. In Kamalika Chaudhuri and Ruslan Salakhutdinov (eds.), *Proceedings of the 36th International Conference on Machine Learning, ICML 2019*, volume 97 of *Proceedings of Machine Learning Research*, pp. 3519–3529, 2019.

Alex Krizhevsky. One weird trick for parallelizing convolutional neural networks. *arXiv preprint arXiv:1404.5997*, 2014.

Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. *Technical report*, 2009.

Yann LeCun, Fu Jie Huang, and Léon Bottou. Learning methods for generic object recognition with invariance to pose and lighting. In *2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2004)*, pp. 97–104, 2004.

Fei-Fei Li, Robert Fergus, and Pietro Perona. One-shot learning of object categories. *IEEE Trans. Pattern Anal. Mach. Intell.*, 28(4):594–611, 2006.

Xuhong Li, Yves Grandvalet, Rémi Flamary, Nicolas Courty, and Dejing Dou. Representation transfer by optimal transport. *arXiv preprint arXiv:2007.06737*, 2020.

Loic Matthey, Irina Higgins, Demis Hassabis, and Alexander Lerchner. dsprites: Disentanglement testing sprites dataset. https://github.com/deepmind/dsprites-dataset/, 2017.

Hossein Mobahi, Mehrdad Farajtabar, and Peter L Bartlett. Self-distillation amplifies regularization in hilbert space. *arXiv preprint arXiv:2002.05715*, 2020.

Gaspard Monge. Mémoire sur la théorie des déblais et des remblais. *Histoire de l'Académie Royale des Sciences de Paris*, 1781.

Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y Ng. Reading digits in natural images with unsupervised feature learning. *NIPS Workshop on Deep Learning and Unsupervised Feature Learning*, 2011.

Maria-Elena Nilsback and Andrew Zisserman. Automated flower classification over a large number of classes. In *Sixth Indian Conference on Computer Vision, Graphics & Image Processing, ICVGIP 2008*, pp. 722–729, 2008.

Omkar M. Parkhi, Andrea Vedaldi, Andrew Zisserman, and C. V. Jawahar. Cats and dogs. In *2012 IEEE Conference on Computer Vision and Pattern Recognition, 2012*, pp. 3498–3505, 2012.

Marco Túlio Ribeiro, Sameer Singh, and Carlos Guestrin. "why should I trust you?": Explaining the predictions of any classifier. In Balaji Krishnapuram, Mohak Shah, Alexander J. Smola, Charu C. Aggarwal, Dou Shen, and Rajeev Rastogi (eds.), *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2016*, pp. 1135–1144, 2016.

Yossi Rubner, Carlo Tomasi, and Leonidas J. Guibas. A metric for distributions with applications to image databases. In *Proceedings of the Sixth International Conference on Computer Vision (ICCV-98)*, pp. 59–66, 1998.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015.

Mark Sandler, Andrew G. Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018*, pp. 4510–4520, 2018.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958, 2014.

Joshua B Tenenbaum, Vin De Silva, and John C Langford. A global geometric framework for nonlinear dimensionality reduction. *science*, 290(5500):2319–2323, 2000.

Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1492–1500, 2017.

Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. In Richard C. Wilson, Edwin R. Hancock, and William A. P. Smith (eds.), *Proceedings of the British Machine Vision Conference 2016, BMVC 2016*, 2016.

Xiaohua Zhai, Joan Puigcerver, Alexander Kolesnikov, Pierre Ruyssen, Carlos Riquelme, Mario Lucic, Josip Djolonga, Andre Susano Pinto, Maxim Neumann, Alexey Dosovitskiy, et al. A large-scale study of representation learning with the visual task adaptation benchmark. *arXiv preprint arXiv:1910.04867*, 2019.

Xian Zhang, Yu Hao, Xiaoyan Zhu, Ming Li, and David R. Cheriton. Information distance from a question to an answer. In Pavel Berkhin, Rich Caruana, and Xindong Wu (eds.), *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2007*, pp. 874–883, 2007.

Xiao Zhang, Xingjian Li, Dejing Dou, and Ji Wu. Measuring information transfer in neural networks. *arXiv preprint arXiv:2009.07624*, 2020.

A TECHNIQUES FOR ACHIEVING PRACTICAL CODELENGTHS

The performance of d_p would rely on how well prequential codelength approximates Kolmogorov complexity. Besides using optimized hyper-parameters when training models in the coding process to minimize codelength, we propose some techniques to enable making more practical estimations. One major limitation in using d_p comes from sampling predictions from f_B: one first need to sample some input x from D (based on data-dependency introduced in Section 2.2). Usually D is unknown but we have a dataset S containing samples from D, so we can use examples from S instead. When the size of S is small, there may not be enough examples to train f_A to converge to f_B. We found that this is often the case for small datasets, for example in Section 4.1. Even when S is large, we can save computation time by using a smaller sample size k.

We introduce some techniques to mitigate the problem of small dataset S:

Re-sampled training. We can use each $x_i \in S$ multiple times by generating m samples from f_B: $y_{i0}, ..., y_{im} \sim f_B(x_i)$. Now (6) becomes

$$\mathbb{E}[L] = \sum_{i=1}^{k} \sum_{j=1}^{m} D_{KL}(f_B(x_i) || f_{\hat{\theta}_{ij}}(x_i)) \tag{15}$$

where $\hat{\theta}_{ij}$ is model trained on all samples before y_{ij}. Because we have more samples from f_B, f_A can be trained to approximate f_B better.

KL-divergence training. More samples can makes training with cross-entropy significantly slower. However, if we let $m \to \infty$, (6) becomes

$$\mathbb{E}[L] = \sum_{i=1}^{k} \sum_{j=1}^{\infty} D_{KL}(f_B(x_i) || f_{\hat{\theta}_{ij}}(x_i)) \tag{16}$$

In practice, (16) will not go to infinity because the output distribution of f is represented with finite precision, D_{KL} will go to zero for sufficiently large m. This also happens in condition of (10). (16) is not practical to compute, but we found in experiments that (16) can be approximated by

$$\mathbb{E}[L] = c \sum_{i=1}^{k} D_{KL}(f_B(x_i) || f_{\hat{\theta}_{i1}}(x_i)) \tag{17}$$

i.e. we only calculate the codelength of the first y sample for each x, and then multiply the coefficient by a coefficient c. In this case, we no longer care about how $\hat{\theta}_{i1}$ is trained on past samples. Because training with an infinite number of samples y_{ij} from $f(x_i)$ using cross-entropy is equivalent to training with KL-divergence with respect to $f(x_i)$, we can use KL-divergence $D_{KL}(f_B(x_i) || f_\theta(x_i))$ directly as the training objective, which makes the training significantly faster. This is similar to training student models in knowledge distillation (Hinton et al., 2015), except that we do not have true labels and only use the KL-divergence term in loss function.

B EXPERIMENT DETAILS

B.1 INVARIANCE EXPERIMENTS

We provide details for modifications used in invariance experiments:

- Scaling: the weights in layer i is multiplied by c, and the weights in layer $i-1$ multiplied by $1/c$. For relu networks, this keeps the output of network unchanged.
- Neuron swapping: we randomly permute c·(total number of neurons) neurons in layer i. We also correspondingly permute the input of layer $i+1$ so that the network output is unchanged.
- Perturbation: we add Gaussian noise of zero mean and standard deviation c to each individual weight of the network.
- Adversarial perturbation: we add a vector v of standard deviation c to the weight vector of the network, and optimize the vector to maximize deviations of second-to-last layer representations. i.e. $max_{v:std(v)=c}\mathbb{E}_x[(f^r_\theta(x) - f^r_{\theta+v}(x))^2]$
- Initialization: we experiment with random initialization and initialize with a pre-trained network on another dataset (CIFAR-10).
- Architecture: we use ResNet architecture with different width and depth. ResNet-56-s refers to ResNet-56 with half the width in each layer.

Next we list the baseline distance (or similarity) measures and how to calculate them:

- Parameter space distances: we concatenate all parameters of the network into a long vector w (excluding parameters in batch normalization layers, because some statistic variable in them can be large and dominate the norm of vector w).
  L2: $||w_A - w_B||_2$.
  Cosine: $w_A \cdot w_B/(||w_A||_2||w_B||_2)$.
  EMD: we use "Optimal Transport of Neurons" in (Li et al., 2020). The distance matrix M is taken to be the pairwise L2 distance between weights of each neuron. The EMD distance is the optimal transport cost of matching neurons from one network to neurons in the other network.
- Representation space distances: we sample x from data distribution D and denote the output of second-to-last layer of model by $f^r(x)$.
  L2: $\mathbb{E}_x[||f^r_A(x) - f^r_B(x)||_2]$.
  Cosine: $\mathbb{E}_x[f^r_A(x) \cdot f^r_B(x)/(||f^r_A(x)||_2||f^r_B(x)||_2)]$.
  EMD: same as in "parameter space distances", but the distance matrix M is taken to be the pairwise L2 distance between the activation vector (on test set) of each neuron.
  Linear CKA: implementation is provided by (Kornblith et al., 2019).
- Output space distances: we use the KL-divergence of output distributions: $\mathbb{E}_x[D_{KL}(f_A(x)||f_B(x))]$,

B.2 GEOMETRY EXPERIMENTS

From the 19 datasets included in VTAB (Zhai et al., 2019), we were able to download 13 datasets for using in this work. Because the dataset size vary greatly among the 13 datasets, we divide them into two groups: larger datasets (size > 10000), which include:

- *cifar100*: CIFAR-100 (Krizhevsky & Hinton, 2009)
- *svhn*: SVHN (Netzer et al., 2011)
- *eurosat*: EuroSAT (Helber et al., 2019)
- *resisc45*: Resisc45 (Cheng et al., 2017)
- *dsprites_position*: dSprites/location (Matthey et al., 2017)
- *dsprites_orientation*: dSprites/orientation

- *smallnorb_azimuth*: SmallNORB/azimuth (LeCun et al., 2004)
- *smallnorb_elevation*: SmallNORB/elevation
- *dmlab*: DMLab (Beattie et al., 2016)

and smaller datasets (size < 10000), which include:

- *caltech101*: Caltech101 (Li et al., 2006)
- *dtd*: DTD (Cimpoi et al., 2014)
- *oxford_flowers102*: Flowers102 (Nilsback & Zisserman, 2008)
- *oxford_iiit_pet*: Pets (Parkhi et al., 2012)

For larger datasets, we use $k = 10000$ as with other experiments. For smaller datasets, we use $k = 2000$. Model is ResNet-56 trained from scratch.

In model geometry experiments, we use the following models trained on ImageNet, as provided by torchvision[1]:

- *resnet18*: ResNet-18 (He et al., 2016)
- *resnet34*: ResNet-34
- *resnet50*: ResNet-50
- *vgg11*: VGG-11 without batch normalization (Simonyan & Zisserman, 2014)
- *vgg11_bn*: VGG-11 with batch normalization
- *alexnet*: AlexNet (Krizhevsky, 2014)
- *resnext50*: ResNeXt-50-32x4d (Xie et al., 2017)
- *wide_resnet50*: WideResNet-50-2 (Zagoruyko & Komodakis, 2016)
- *densenet121*: Densenet-121 (Huang et al., 2017)
- *squeezenet*: SqueezeNet 1.1 (Iandola et al., 2016)
- *mobilenet*: MobileNet V2 (Sandler et al., 2018)

Codelength is calculated on the training set of ILSVRC 2012 (Russakovsky et al., 2015).

B.3 Ensemble experiments

We train multiple ResNet-56 models with 2 different random initializations and half of the examples sampled form the TinyImageNet[2] training set. This means the training examples seen by each model can overlap from 0% to 100%. We then select two models out of this collection and ensemble the two models. Ensemble performance and distance between the two models measured by $d_p(f_A, f_B)$ is given in Table 1. We list pairs with same or different initialization, and with different overlap in training examples. Generally speaking, the less the training examples overlaps the larger the distance between the models. Different initialization can also makes the models more dissimilar. Note that from Figure 6, we see that distance $d_p(f_A, f_B)$ is correlated with ensemble performance regardless of whether diversity comes from difference in training examples or difference in initializations.

B.4 Generalization experiments

We run experiments on CIFAR-10 with different hyperparameters and model configurations, and in all configurations we train the model to cross-entropy loss of 0.1 on training set. Then we measure the generalization gap as the loss on testing set minus the loss on training set. Starting from a default configuration (which is the same hyperparameters we use in other experiments in this paper), each time we modify one of the hyperparameters. Results are listed in Table 2. In terms of studying generalization gap, our experiments is far less thorough than in (Jiang et al., 2020), but here we would like to illustrate the connection between d_p with generalization gap under different experiment settings, without spending too many machine hours.

[1] https://github.com/pytorch/vision

[2] https://tiny-imagenet.herokuapp.com

Table 1: Details of configurations in ensemble experiments

Initialization seed	Training set overlap	Ensemble loss	Ensemble accuracy	$d_p(f_A, f_B)$
same	0%	2.197	47.22	2502
same	25%	2.211	46.59	2234
same	50%	2.229	46.40	1946
same	75%	2.229	45.93	1569
different	0%	2.192	47.09	2957
different	25%	2.201	47.09	2481
different	50%	2.214	47.06	2195
different	75%	2.235	45.99	1772
different	100%	2.257	45.75	1436

Table 2: Details of configurations in generalization experiments.

Configuration	Architecture	Optimizer	Batch size	Learning rate	Data augmentation	Generalization gap	$d_p(f, \mathbf{0})$
default	ResNet-56	Adam	32	0.002	off	0.32	4605
lr=0.0002	ResNet-56	Adam	32	**0.0002**	off	0.80	3045
lr=0.02	ResNet-56	Adam	32	**0.02**	off	0.77	3322
bs=8	ResNet-56	Adam	**8**	0.002	off	0.50	3908
bs=128	ResNet-56	Adam	**128**	0.002	off	0.69	3220
opt=SGD	ResNet-56	**SGD**	32	0.2	off	0.68	3344
arch=resnet20	**ResNet-20**	Adam	32	0.002	off	0.79	3348
arch=LeNet	**LeNet-5**	Adam	32	0.002	off	0.84	2873
data_aug=on	ResNet-56	Adam	32	0.002	**on**	0.62	3615