NEURAL NETWORKS BEHAVE AS HASH ENCODERS: AN EMPIRICAL STUDY

Anonymous authors
Paper under double-blind review

ABSTRACT

The input space of a neural network with ReLU-like activations is partitioned into multiple linear regions, each corresponding to a specific activation pattern of the included ReLU-like activations. We demonstrate that this partition exhibits the following encoding properties across a variety of deep learning models: (1) *determinism*: almost every linear region contains at most one training example. We can therefore represent almost every training example by a unique activation pattern, which is parameterized by a *neural code*; and (2) *categorization*: according to the neural code, simple algorithms, such as K-NN and logistic regression, can achieve fairly good performance on both training and test data. These encoding properties surprisingly suggest that *normal neural networks well-trained for classification behave as hash encoders without any extra efforts.* In addition, the encoding properties exhibit variability in different scenarios. Further experiments demonstrate that *larger model capacity, longer training time, and more training data lead to stronger encoding properties, while regularization has a relatively little impact.* We then define an *activation hash phase chart* to represent the space expanded by model capacity, training time, training sample size, and the encoding properties, which is divided into three canonical regions: *under-expressive regime*, *critically-expressive regime*, and *sufficiently-expressive regime*.

1 INTRODUCTION

Recent studies have highlighted that the input space of a rectified linear unit (ReLU) network is partitioned into linear regions by the nonlinearities in the activations (Pascanu et al., 2013; Montufar et al., 2014; Raghu et al., 2017), where ReLU networks refer to the networks with only ReLU-like (two-piece linear) activation functions (Glorot et al., 2011; Maas et al., 2013; He et al., 2015; Arjovsky et al., 2016). Specifically, the mapping induced a ReLU network is linear with respect to the input data within linear regions and nonlinear and non-smooth in the boundaries between linear regions. Intuitively, the interiors of linear regions correspond to the linear parts of the ReLU activations and thus corresponds to a specific activation pattern of the ReLU-like activations, while the boundaries are induced by the turning points. Therefore, every example can be represented by the corresponding activation pattern of the linear region where it falls in. In this paper, we parameterize the activation pattern as a 0-1 matrix, which is termed as *neural code*. Correspondingly, a neural network induces an *activation mapping* from every input example to its neural code. For the detailed definition of neural code, please refer to Section 3. This linear region partition can be extended to the neural networks containing smooth activations (such as sigmoid activations and tanh activations) as well as ReLU-like activations, in which the interiors are no longer linear but still smooth.

Through a comprehensive empirical study, this paper shows:

> *A well-trained normal neural network performs a hash encoder without any extra effort, where the neural code is the hash code and the activation mapping is the hash function.*

Specifically, our experiments demonstrate that the neural code exhibits the following encoding properties shared by hash code (Knuth, 1998) in most common scenarios of deep learning for classification tasks:

- **Determinism:** When a neural network has been well trained, the overwhelming majority of the linear regions contain at most one training example per region. Thus, almost every training example can be represented by a unique neural code. To evaluate this determinism property quantitatively, we define a new term *redundancy ratio*, which is the proportion of the training examples that share activations with others. Experimental results show that the redundancy ratio is near zero in almost every scenario.

- **Categorization:** The neural codes of examples from the same category are close to each other in the neural code space under the distance whereon (such as Euclidean distance and Hamming distance), while the neural codes are far away from each other if the corresponding examples are from different categories. We conduct clustering and classification experiments on the neural code space. Empirical results suggest that simple algorithms, such as K-means (Lloyd, 1982), K-NN (Cover & Hart, 1967; Duda et al., 1973), and logistic regression can achieve fairly good training and test performance.

It is worth noting that our study is different to the efforts of employing neural networks to learn hash functions, where the outputs are the hash codes of the input examples (Wang et al., 2017), while this paper reports that the activation pattern (or neural code) is already fairly good hash code. Specifically, Xia et al. (2014); Lai et al. (2015); Zhu et al. (2016); Cao et al. (2017; 2018) designed hash layers to neural network for learning hash functions of images; and Simonyan & Zisserman (2014); Donahue et al. (2015); Wang et al. (2016); Varol et al. (2017); Chao et al. (2018); Yuan et al. (2019) extended the applicable domain to video data. Surprisingly, our experiments demonstrate that well-trained normal neural networks used for classification already induce fairly good hash encoders.

The encoding properties also exhibit some variabilities in different scenarios. We then conduct comprehensive experiments to investigate which factors would influence the encoding properties. The empirical results suggest that three factors play major roles in shaping the encoding properties: *model capacity*, *training time*, *and sample size*, while regularization has a relative little impact. Specifically, larger model capacity, longer training time, and more training data lead to stronger encoding properties. The two encoding properties collectively measure the expressivity of the activation mapping. For the brevity, we term this expressivity as *goodness-of-hash*.

We evertually define an *activation hash phase chart* to characterize the space expanded by model capacity, training time, sample size, and the goodness-of-hash. According to the discovered correlations, this space is partitioned into three canonical regions:

- **Under-expressive regime.** The redundancy ratio is considerably higher than zero while the categorization accuracy is considerably lower than 100%. However, both redundancy ratio and categorization accuracy exhibit significant *positive* correlations with model capacity, training time, and training sample size.

- **Critically-expressive regime.** This is a transition region between the under-expressive and sufficiently-expressive regimes. The goodness-of-hash changes considerably as model capacity, training time, and sample size change, while the correlations become insignificant.

- **Sufficiently-expressive regime.** The redundancy ratio is almost zero while the categorization accuracy has become fairly good. We can hardly observe them change when model capacity, training time, and training sample size change. This regime involves almost all common scenarios in the current practice of deep learning.

It is worths noting that our partition is different from the one proposed by Nakkiran et al. (2019), which characterizes the the expressivity or expressive power of the input-output mapping induced by a neural network. By contrast, our the partition in activation hash phase chart characerizes goodness-of-hash.

Our results are established on empirical results of multi-layer perceptrons (MLPs) (Rosenblatt, 1961), VGG (Simonyan & Zisserman, 2015), ResNets (He et al., 2016a;b), ResNeXt (Xie et al., 2017), and DenseNet (Huang et al., 2017) for classification on the dataset MNIST (LeCun et al., 1998). The training code, well-trained models, and obtained raw data will be released on GitHub.

2 RELATED WORKS

Many works have also studied the number of linear regions (linear region counting) in neural networks containing ReLU activations. Pascanu et al. (2013); Montufar et al. (2014) proposed an upper bound that is exponential with respect to the network's depth and polynomial with respect to the width. Hanin & Rolnick (2019b) Montufar et al. (2014); Arora et al. (2016) also proved lower bounds for the linear region counting. Additionally, Poole et al. (2016); Novak et al. (2018); Hanin & Rolnick (2019a) investigated how the linear region counting would change as the training progresses. Zhang & Wu (2020) reported that optimization methods would also significantly influence the geometry property of linear regions. Other advances on the linear region counting include Bianchini & Scarselli (2014); Raghu et al. (2017); Charisopoulos & Maragos (2018); Hu & Zhang (2018); Serra et al. (2018); Kumar et al. (2019); Phuong & Lampert (2019); Inoue (2019); Xiong et al. (2020); Zhu et al. (2020).

A partition in the loss surfaces of neural networks has also been observed. Soudry & Hoffer (2018) highlighted that the loss surfaces of neural networks with piecewise linear functions are partitioned into multiple smooth and multilinear open cells, while the boundaries are non-differentiable. He et al. (2020) discovered three other properties: (1) every local minimum in a cell is the global minimum in the cell; (2) local minima in a cell are interconnected; and (3) all local minima in a cell are in an equivalence class. This paper focuses on another partition observed in the data space.

3 PRELIMINARIES

Suppose a ReLU network $\mathcal{N}$ is trained to fit a dataset $S = \{(x_i, y_i), i = 1, \ldots, n\}$ for classification, where $x_i \in \mathcal{X} \subset \mathbb{R}^{d_X}$, d_X is the dimension of x, $y_i \in \mathcal{Y} = \{1, \ldots, d\}$, d is the number of potential categories, and n is the training sample size. For the brevity, we may also rewrite an example as $z_i = (x_i, y_i) \in \mathcal{Z} = \mathcal{X} \times \mathcal{Y}$ in the rest of this paper. Additionally, we assume that all examples z_i are independent and identically distributed (i.i.d.) random variables drawn from a data distribution $\mathcal{D}$. Eventually, we denote the well-trained model as $\mathcal{M}$.

Recent works have shown that the input space of a ReLU network $\mathcal{N}$ is partitioned into multiple linear regions, each of which corresponds to a specific activation pattern of the ReLU activation functions. In this paper, we represent the activation pattern as an $l \times w$ matrix $\boldsymbol{P} \in \mathcal{P}$, where l and w are the depth and the largest width of this neural network $\mathcal{N}$, respectively. Specifically, the (i, j)-th component characterizes the activation statue of the j-th ReLU neuron in i-th layer, 1 represents that this neuron is activated, 0 means this neuron is deactivated or invalid. Different layers may have different numbers of neurons. Therefore, there might be some indices (i, j) are invalid. We represent the activation patterns of these neurons as 0 since they are never activated. The matrix $\boldsymbol{P}$ is named as *neural code*. We can also formulate the neural code as a vector if no confusion of the depth and width occurs.

It is worth noting that the volume of boundaries between linear regions is zero, because the boundaries correspond to at least one turning point in the activations, which are of measure zero. Correspondingly, the probability that some examples fall in the boundaries is zero. We thus assume no example is in the boundaries. Therefore, fixing the weight w of the model $\mathcal{M}$, every example $x \in \mathcal{X}$ can be indexed by the neural code $\boldsymbol{P}$ of the corresponding linear region. It is worth noting that the instance x can be either seen in the training sample set or the test sample set.

4 NEURAL NETWORKS PERFORM AS HASH ENCODERS

Through an empirical study, this paper discovers that (1) the neural code is the hash code of the corresponding datum; and (2) correspondingly, the mapping $\mathcal{X} \to \mathcal{P}$ from datums to their neural codes is a hash function, which is termed as *activation mapping*. In contrast to the learning-to-hash methods, we find well-trained normal neural networks for normal tasks (such as classification) already perform hash encoding without any extra efforts. These results are verified empirically in a variety of common scenarios in deep learning. Specifically, the neural code exhibits two major *encoding properties*: (1) *determinism:* the overwhelming majority of the linear regions contain at most one training example per region. Thus, almost every training example can be represented by a unique neural code; and (2) *categorization:* the neural codes of examples from the same category are close

to each other in the neural code space under the distance whereon (such as Euclidean distance and Hamming distance), while the neural codes are far away from each other if the corresponding examples are from different categories. These two properties collectively characterise the expressivity of the activation mapping, which is termed as *goodness-of-hash*.

Similar to the works in learning-to-hash, we adopt the following two measures to quantitatively evaluate the activation mapping as a hash mapping.

- **Redundancy ratio.** Some instances x may fall in the same linear region, and therefore they share the same neural code. For a sample set, we can calculate the proportion of the instances that share linear regions with others. This proportion is termed as *redundancy ratio* in this paper. Generally, a larger redundancy ratio corresponds a worse encoding property.
- **Categorization accuracy.** Hash code is usually employed for nearest neighbor searching. In this paper, we perform simple clustering and classification algorithms, such as K-means and K-NN, to the neural codes. We employ the training and test accuracy to evaluate the encoding property. Specifically, higher accuracy corresponds to a better encoding property.

We empirically investigate the activation mapping of several MLPs, VGG, ResNet-18, ResNet-34, ResNeXt-26, DenseNet-28 on the MNIST dataset. We calculate the redundancy ratio and conduct K-means and K-NN in these scenarios, which are presented in Table 1. The results clearly demonstrate that the redundancy ratio is almost zero and the categorization accuracy is fairly good in almost every scenario, and thus verify the determinism and categorization properties. Moreover, we visualized the neural codes employing t-SNE (Maaten & Hinton, 2008) in Figure 1. This qualitative result demonstrates that examples from the same category are concentrated together while clear boundaries are observed between examples from different categories, which are in full agreement with the two encoding properties. More details of the experiment implementations are omitted from the main text and given in Appendix A due to the space limitation.

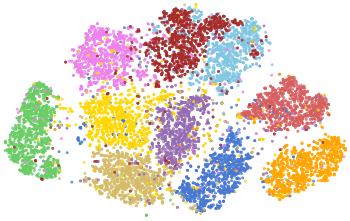


Figure 1: t-SNE visualization of the neural codes of MNIST generated by a one-hidden-layer MLP with width 100. The MLP is trained on MNIST for classification.

Table 1: Test accuracy of K-means (based on Euclidean distance) and K-NN (based on Hamming distance) on the neural code generared from MNIST by VGG-18, ResNet-18, ResNet-34, ResNeXt-26, and DenseNet-28. All models are trained on MNIST.

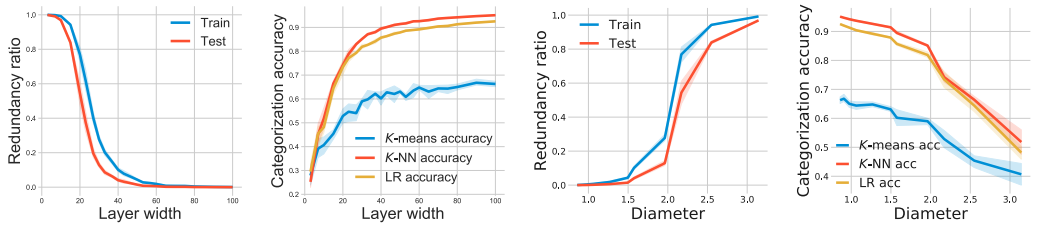
Architecture	K-means Accuracy	K-NN Accuracy
VGG-18	99.95%	99.33%
ResNet-18	98.96%	99.32%
ResNet-34	99.66%	99.49%
ResNeXt-26	98.31%	99.24%
DenseNet-28	69.87%	98.59%

5 Factors that Shape of Encoding Properties

The results presented in Table 1 also suggest that the encoding properties exhibit variability in different scenarios. Through comprehensive experiments, we investigate which factors would influence the encoding properties. The investigated factors include model capacity, training time, sample size, and several popular regularization techniques. Some details of the experiment implementations are omitted from the main text and given in Appendix A due to the space limitation.

5.1 Relationship between Model Capacity and Encoding Properties

We first investigate how model capacity would influence the encoding properties. The experiments are conducted based on one-hidden-layer MLPs on the MNIST dataset. Each experiment is run for 5 trials.



(a) Determinism vs. LW (b) Categorization vs. LW (c) Determinism vs. D (d) Categorization vs. D

Figure 2: (a) Plots of redundancy ratio as a function of the layer width (LW) of MLPs on both training set (blue) and test set (red) of MNIST. (b) Plots of test accuracies of K-means (blue), K-NN (red), and logistic regression (LR, range) as functions of the layer widths (LW) of MLPs on MNIST. (c) Plots of redundancy ratio as a function of the diameter (D) of MLPs on both training set (blue) and test set (red) of MNIST. (d) Plots of test accuracies of K-means (blue), K-NN (red), and logistic regression (LR, range) as functions of the diameter (D) of MLPs on MNIST. All models are trained on MNIST for classification for 5 times with different random seeds. The darker lines show the average over seeds and the shaded area shows the standard deviations.

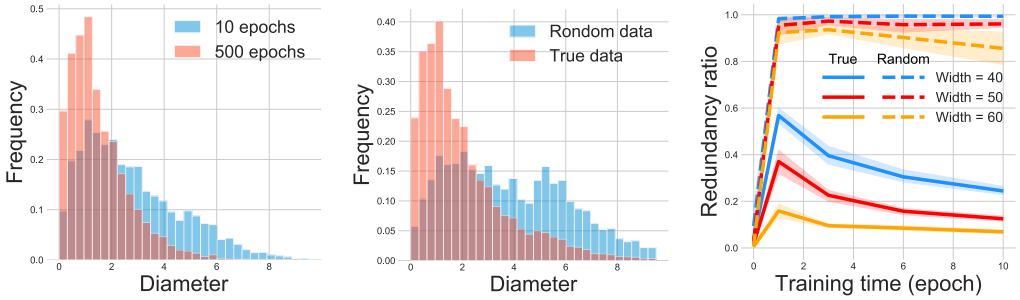
Measure model capacity by width. For one-hidden-layer MLPs, a natural measure for the model capacity is the layer width. Thus, we train MLPs with multiple widths (please see the full list of widths involved in our experiments in Appendix A) till the model has converged to a steady state, while all irrelative variables are strictly controlled. Every well-trained model generates an activation mapping, which is then employed to map all examples in both the training and test sample sets to their neural codes. We then calculate the redundancy ratios of the neural codes and conducted K-means, $K-$NN, and logistic regression algorithms based on the neural codes. The results are illustrated in Figure 2(a) and Figure 2(b).

From the plots, we can observe clear correlations between the encoding properties and the width: (1) the redundancy ratio starts at around 1 when the width is 3. Then, the redundancy ratio decreases to near zero on both datasets as the width increases; and (2) the accuracies of K-means, K-NN, and logistic regression start at about 25% when the width is 3. Then, as the width increases, the K-means accuracy increases to around 70% and the test accuracy of K-NN and logistic regression increase to larger than 90%.

Measure model capacity by the diameters of linear regions. Another diameters of linear regions. However, calculating diameters directly would be extremely time consuming, especially when the dimension of the data space is high. To address this issue, we propose an *average stochastic activation diameter* as a new measure for evaluating the model capacity, which is calculated via three steps: (1) for every linear region containing an example, we sample a random direction via the uniform distribution on the set of all potential directions; (2) we calculate the diameter of the activation in this direction, which is defined to be the longest intersected line segment of the linear region and a line along the sampled direction. The length of this intersected line segment is termed as the *stochastic diameter*; and (3) we calculate the mean of the stochastic diameters of all the linear regions, which is termed as the average stochastic activation diameter. Intuitively, a smaller average stochastic activation diameter means that the input space has been divided into smaller linear regions, and thus can represent more sophisticated data structures. Therefore, can serve as a measure of model capacity. Correspondingly, a negative correlation between the average stochastic activation diameter and the encoding properties is expected, as illustrated in Figure 2(c) and Figure 2(d).

We also tested the redundancy ratio beyond the data distribution of the training data. A set of examples is generated according to the uniform distribution over the unit ball centered at the original point. We observe that the redundancy ratio is larger than 0.8 on the randomly generated data; see Figure 3(c). This result suggests that the determinism property no longer stands, and correspondingly, one cannot represent randomly generated data by unique neural codes. Therefore, the categorization property also becomes elusive. We then summarize these results as the following remark.

Remark 1. *The encoding properties would not apply beyond the training data distribution.*



(a) Diameters in different time (b) Diameters for random/true data (c) Determinism vs. time

Figure 3: (a) Histograms of diameters of stochastic diameters calculated on MNIST for an MLP of width 50 trained on MNIST for 10 epochs (red) and 500 epochs (blue), respectively. (b) Histograms of stochastic diameters calculated on MNIST (blue) and randomly generated data with the same dimension (red), respectively, for an MLP of width 50 trained on MNIST. (c) Plots of redundancy ratio calculated on MNIST ("True data", solid lines) and randomly generated data (dotted lines) as functions of training time for MLPs of widths 40 (blue), 50 (red), and 60 (orange). All models are trained on MNIST for classification for 5 times with different random seeds. The darker lines show the average over seeds and the shaded area shows the standard deviations.

We further propose the following hypothesis to explain this remark:

Hypothesis 1. *The diameters of linear regions in the support of data distribution becomes smaller as the training progresses, while the diameters of regions far away do not change much.*

We then collect the average stochastic activation diameters for each scenario, as illustrated in Figure 3(a) and Figure 3(b). We observe that diameter distribution is more concentrated at a lower value with lager training epoch; see Figure 3(a). Moreover, we observe an interesting result that diameter distribution with true examples is more concentrated at a lower value than diameter distribution with random examples. As an example, we draw a diagram of diameter distribution; see Figure 3(b). The diagrams for other scenarios are given in Appendix B. These results are in full agreement with our hypothesis.

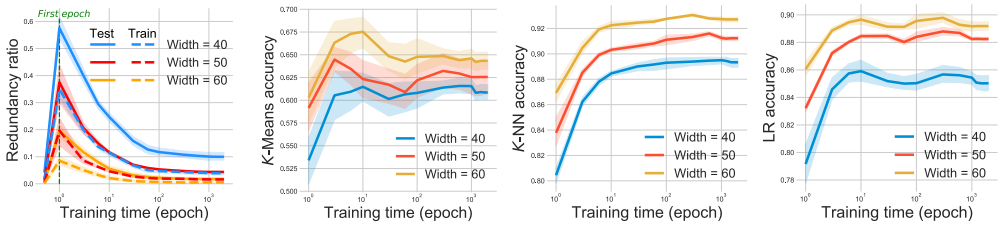
5.2 Relationship between training time and encoding properties

We next investigate the influence of training time on the encoding properties. We trained 210 one-hidden-layer MLPs with widths of 40, 50, and 60 on the MNIST dataset. Each scenario is repeated for 5 trials.

We collect the redundancy ratio and the categorization accuracy of every epoch in all the scenarios aforementioned, on which encoding properties curves are drawn, as shown in Figures 4. The plots clearly suggest a positive correlation between the training time and the encoding properties: (1) the redundancy ratio is initial large (about 0.20 for width 40, 0.47 for width 50, and 0.57 for width 60); see Figure 4(a). Then, as the training procedure progresses, the redundancy ratio decreases monotonically to zero when after approximately 100 epochs; and (2) the accuracies of K-means, K-NN, and logistic regression start at around zero, then monotonically increase significantly as the training progresses, and eventually to around 100% after approximately 10 training epochs.

5.3 Relationship between sample size and encoding properties

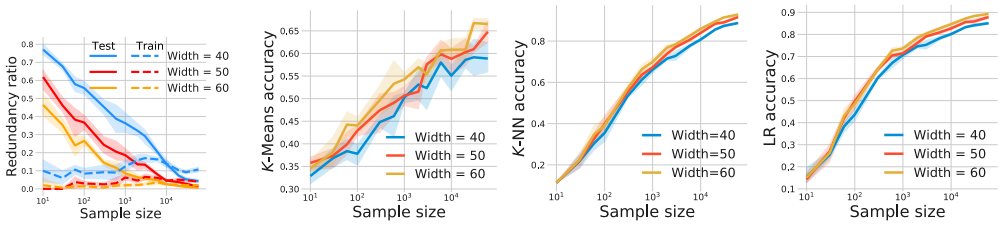
We then investigate how sample size impacts encoding properties. We trained 210 MLPs with width 40, 50, and 60 on training sample sets of different sizes, while all irrelevant variables are strictly controlled. Specifically, the training sample sets are randomly drawn from of MNIST. It is worth noting that we adopt the number of iterations rather than the number of epochs in measuring the training time because the number of iterations in one epoch grows proportionally with the sample size increases.



(a) Determinism vs. time

(b) Test accuracy of K-means, K-NN, and logistic regression vs. time

Figure 4: (a) Redundancy ratio as a function of training time. (b) Test accuracy of K-means (left), K-NN (middle), and logistic regression (right) as functions of training time. The models are MLPs of depths 40 (blue), 50 (red), and 60 (orange). All models are trained on MNIST for classification for 5 times with different random seeds. The darker lines show the average over seeds and the shaded area shows the standard deviations.



(a) Determinism vs. sample size on both training and test sets

(b) Test accuracy of K-means, K-NN, and logistic regression vs. sample size

Figure 5: (a) Redundancy ratios on training set (dotted lines) and test set (solid lines) of MNIST as functions of sample size. (b) Test accuracy of K-means (left), K-NN (middle), and logistic regression (right) as functions of sample size. The models are MLPs of depths 40 (blue), 50 (red), and 60 (orange). All models are trained on MNIST for classification for 5 times with different random seeds. The darker lines show the average over seeds and the shaded area shows the standard deviations.

We calculated the redundancy ratio of the neural code and conducted K-means, K-NN, and logistic regression algorithms on the neural code on both the training and test sample sets in all scenarios; see the redundancy ratios in Figure 5(a) and the accuracies of K-means, K-NN, logistic regression in Figure 5(b). From the plots, two main observations are obtained: (1) the redundancy ratio calculated on either the training sample set or the test sample starts at a considerably high position at initialization, and then decreases monotonically to near zero as the training sample size increases; and (2) the test accuracies of K-means, K-NN, and logistic regression have clear positive correlations with the sample size: the K-means accuracy increases from 35% to 60%, the K-NN accuracy increases from 10% to 90%, and the logistic regression accuracy increases from 10% to 85%, respectively, as the sample size increases.

5.4 Impact of Regularization

We also examine the impact of regularization on the encoding properties. Specifically, we trained 345 MLPs on the MNIST dataset. The results suggest that the impact of regularization is relatively small; see Figure 6. We only present results for batch normalization here due to space limitation, which is the most significant regularization technique we have tested. More data for gradient clipping and weight decay are shown in Appendix B.

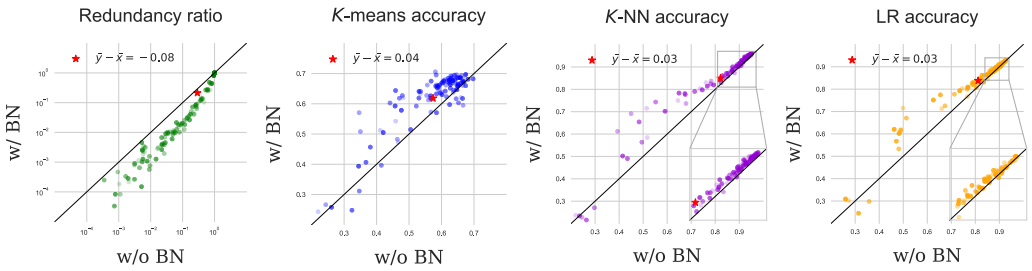


Figure 6: Scatter of redundancy ratios and test accuracy of K-means (blue), K-NN (violet), and logistic regression (LR, orange) for MLPs of depth from 3 to 100 with batch normalization (w/ BN, y-axis) and without batch normalization (w/o BN, x-asix). We perfer a smaller $\bar{y} - \bar{x}$ in redundancy ration, and larger ones in the test accuracies of K-means, K-NN, and logistic regression. Totally, 115 models are involved in one scatter.

5.5 Activation hash phase chart

We eventually can define an *activation hash phase chart* that characterizes the space expanded by redundancy ratio, categorization accuracy, model capacity, training time, and sample size. Summarizing the relationships discovered above, the activation hash phase chart is divided into three canonical regions:

- **Under-expressive regime.** The encoding properties are much lower than desired. The redundancy ratio is considerably greater than zero, while the categorization accuracies are considerably lower than 100%. However, the encoding properties exhibit significant positive correlations with model capacity, training time, and training sample size.
- **Critically-expressive regime.** This is a turning region between the under-expressive regime and sufficiently-expressive regime. Both redundancy ratio and clustering may change considerably as model capacity, training time, and sample size change, while the correlations become unsignificant.
- **Sufficiently-expressive regime.** The redundancy ratio is almost zero while the categorization accuracies are nearly 100%. These hardly change when model capacity, training time, and sample size increase. This regime covers most common scenarios in deep learning.

6 Conclusion

This paper studies the linear partition in the linear spaces of neural networks with ReLU-like activations. In this partition, every region corresponds to an activation pattern of the ReLU-like activations, which is parameterized by *neural code* in this paper. We discover that the neural code behaves as the hash code of the corresponding example. Specifically, the neural code possesses the following encoding properties: (1) *determinism*: almost every linear region contain at most one example. Correspondingly, almost every example can be represented by one unique neural code. This property can be quantitatively evaluated by *redundancy ratio*, which is defined to be the proportion of the examples sharing a neural code with others; and (2) *categorization*: simple classification and clustering algorithms, such as K-NN, logistic regression, and K-means can achieve fairly good training accuracy and test accuracy on the neural code space. These properties also exhibit variabilities in different scenarios. We then find three main factors, *model capacity*, *training time*, and *sample size*, play major roles in shaping the encoding properties, while several popular regularizers have a little impact. Specifically, larger model capacity, longer training time, and more training data correspond to stronger encoding properties. Accordingly, we define an *activation hash phase chart* to represent the space spanned by model capacity, training time, sample size, and the encoding properties, which is divided into three canonical regions: under-expressive regime, critically-expressive regime, and sufficiently-expressive regime.

REFERENCES

Martin Arjovsky, Amar Shah, and Yoshua Bengio. Unitary evolution recurrent neural networks. In *International Conference on Machine Learning*, pp. 1120–1128, 2016.

Raman Arora, Amitabh Basu, Poorya Mianjy, and Anirbit Mukherjee. Understanding deep neural networks with rectified linear units. *arXiv preprint arXiv:1611.01491*, 2016.

Monica Bianchini and Franco Scarselli. On the complexity of neural network classifiers: A comparison between shallow and deep architectures. *IEEE transactions on neural networks and learning systems*, 25(8):1553–1565, 2014.

Yue Cao, Mingsheng Long, Bin Liu, and Jianmin Wang. Deep cauchy hashing for hamming space retrieval. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1229–1237, 2018.

Zhangjie Cao, Mingsheng Long, Jianmin Wang, and Philip S Yu. Hashnet: Deep learning to hash by continuation. In *Proceedings of the IEEE international conference on computer vision*, pp. 5608–5617, 2017.

Yu-Wei Chao, Sudheendra Vijayanarasimhan, Bryan Seybold, David A Ross, Jia Deng, and Rahul Sukthankar. Rethinking the faster r-cnn architecture for temporal action localization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1130–1139, 2018.

Vasileios Charisopoulos and Petros Maragos. A tropical approach to neural networks with piecewise linear activations. *arXiv preprint arXiv:1805.08749*, 2018.

Thomas Cover and Peter Hart. Nearest neighbor pattern classification. *IEEE transactions on information theory*, 13(1):21–27, 1967.

Jeffrey Donahue, Lisa Anne Hendricks, Sergio Guadarrama, Marcus Rohrbach, Subhashini Venugopalan, Kate Saenko, and Trevor Darrell. Long-term recurrent convolutional networks for visual recognition and description. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2625–2634, 2015.

Richard O Duda, Peter E Hart, and David G Stork. *Pattern classification and scene analysis*, volume 3. Wiley New York, 1973.

Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Deep sparse rectifier neural networks. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pp. 315–323, 2011.

Boris Hanin and David Rolnick. Complexity of linear regions in deep networks. In *International Conference on Machine Learning*, pp. 2596–2604, 2019a.

Boris Hanin and David Rolnick. Deep relu networks have surprisingly few activation patterns. In *Advances in Neural Information Processing Systems*, pp. 361–370, 2019b.

Fengxiang He, Bohan Wang, and Dacheng Tao. Piecewise linear activations substantially shape the loss surfaces of neural networks. *arXiv preprint arXiv:2003.12236*, 2020.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pp. 1026–1034, 2015.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Conference on Computer Vision and Pattern Recognition*, 2016a.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. In *European conference on computer vision*, 2016b.

Qiang Hu and Hao Zhang. Nearly-tight bounds on linear regions of piecewise linear neural networks. *arXiv preprint arXiv:1810.13192*, 2018.

Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4700–4708, 2017.

Kenta Inoue. Expressive numbers of two or more hidden layer relu neural networks. In *2019 Seventh International Symposium on Computing and Networking Workshops (CANDARW)*, pp. 129–135. IEEE, 2019.

Donald E Knuth. *The art of computer programming: Volume 3: Sorting and Searching*. Addison-Wesley Professional, 1998.

Abhinav Kumar, Thiago Serra, and Srikumar Ramalingam. Equivalent and approximate transformations of deep neural networks. *arXiv preprint arXiv:1905.11428*, 2019.

Hanjiang Lai, Yan Pan, Ye Liu, and Shuicheng Yan. Simultaneous feature learning and hash coding with deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3270–3278, 2015.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Stuart Lloyd. Least squares quantization in pcm. *IEEE transactions on information theory*, 28(2): 129–137, 1982.

Andrew L Maas, Awni Y Hannun, and Andrew Y Ng. Rectifier nonlinearities improve neural network acoustic models. In *International Conference on Machine Learning*, pp. 3, 2013.

Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605, 2008.

Guido F Montufar, Razvan Pascanu, Kyunghyun Cho, and Yoshua Bengio. On the number of linear regions of deep neural networks. In *Advances in neural information processing systems*, pp. 2924–2932, 2014.

Preetum Nakkiran, Gal Kaplun, Yamini Bansal, Tristan Yang, Boaz Barak, and Ilya Sutskever. Deep double descent: Where bigger models and more data hurt. *arXiv preprint arXiv:1912.02292*, 2019.

Roman Novak, Yasaman Bahri, Daniel A Abolafia, Jeffrey Pennington, and Jascha Sohl-Dickstein. Sensitivity and generalization in neural networks: an empirical study. *arXiv preprint arXiv:1802.08760*, 2018.

Razvan Pascanu, Guido Montufar, and Yoshua Bengio. On the number of response regions of deep feed forward networks with piece-wise linear activations. *arXiv preprint arXiv:1312.6098*, 2013.

Mary Phuong and Christoph H Lampert. Functional vs. parametric equivalence of relu networks. In *International Conference on Learning Representations*, 2019.

Ben Poole, Subhaneil Lahiri, Maithra Raghu, Jascha Sohl-Dickstein, and Surya Ganguli. Exponential expressivity in deep neural networks through transient chaos. In *Advances in neural information processing systems*, pp. 3360–3368, 2016.

Maithra Raghu, Ben Poole, Jon Kleinberg, Surya Ganguli, and Jascha Sohl-Dickstein. On the expressive power of deep neural networks. In *international conference on machine learning*, pp. 2847–2854. PMLR, 2017.

Frank Rosenblatt. Principles of neurodynamics. perceptrons and the theory of brain mechanisms. Technical report, Cornell Aeronautical Lab Inc Buffalo NY, 1961.

Thiago Serra, Christian Tjandraatmadja, and Srikumar Ramalingam. Bounding and counting linear regions of deep neural networks. In *International Conference on Machine Learning*, pp. 4558–4566. PMLR, 2018.

Karen Simonyan and Andrew Zisserman. Two-stream convolutional networks for action recognition in videos. In *Advances in neural information processing systems*, pp. 568–576, 2014.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations*, 2015.

Daniel Soudry and Elad Hoffer. Exponentially vanishing sub-optimal local minima in multilayer neural networks. In *International Conference on Learning Representations Workshop*, 2018.

Gül Varol, Ivan Laptev, and Cordelia Schmid. Long-term temporal convolutions for action recognition. *IEEE transactions on pattern analysis and machine intelligence*, 40(6):1510–1517, 2017.

Jingdong Wang, Ting Zhang, Nicu Sebe, Heng Tao Shen, et al. A survey on learning to hash. *IEEE transactions on pattern analysis and machine intelligence*, 40(4):769–790, 2017.

Limin Wang, Yuanjun Xiong, Zhe Wang, Yu Qiao, Dahua Lin, Xiaoou Tang, and Luc Van Gool. Temporal segment networks: Towards good practices for deep action recognition. In *European conference on computer vision*, pp. 20–36. Springer, 2016.

Rongkai Xia, Yan Pan, Hanjiang Lai, Cong Liu, and Shuicheng Yan. Supervised hashing for image retrieval via image representation learning. In *AAAI*, volume 1, pp. 2, 2014.

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.

Huan Xiong, Lei Huang, Mengyang Yu, Li Liu, Fan Zhu, and Ling Shao. On the number of linear regions of convolutional neural networks. *arXiv preprint arXiv:2006.00978*, 2020.

Li Yuan, Eng Hock Francis Tay, Ping Li, and Jiashi Feng. Unsupervised video summarization with cycle-consistent adversarial lstm networks. *IEEE Transactions on Multimedia*, 2019.

Xiao Zhang and Dongrui Wu. Empirical studies on the properties of linear regions in deep neural networks. *arXiv preprint arXiv:2001.01072*, 2020.

Han Zhu, Mingsheng Long, Jianmin Wang, and Yue Cao. Deep hashing network for efficient similarity retrieval. In *Thirtieth AAAI Conference on Artificial Intelligence*, 2016.

Rui Zhu, Bo Lin, and Haixu Tang. Bounding the number of linear regions in local area for neural networks with relu activations. *arXiv preprint arXiv:2007.06803*, 2020.