Neural scaling law from toy model of hierarchical data

Anonymous Authors

Abstract

In our investigation, we explore the phenomenon known as the 'Neural scaling law,' which describes a power-law-like relationship between test error and the number of training points in neural networks trained on real data. This behavior is believed to arise from the division of the task into numerous sub-tasks, each learned at different scales of training points. In our study, we construct a toy model with a hierarchical tree structure , termed the Random Hierarchy Model. We propose that in our case, subtask is associated to identifying a generative path. We impose for every layer in hierarchy, each feature is mapped to lower level tuple where frequency follows zipf law with exponent $-(1+\alpha)$. Then the frequency of generative path follows zipf law with exponent $-(1+\alpha')$, where we can derive it from α. In this case, there is natural critical value of α where $\alpha' = 0$, and we denote it α_c. Empirically, when $\alpha > \alpha_c$ (i.e. $\alpha' > 0$), accuracy of test error is related to how many paths of each datum is seen in training set, so if more paths are seen in training set, accuracy is higher. Also test error roughly follows power law with exponent $-\frac{\alpha'}{1+\alpha'}$. However when $\alpha < \alpha_c$ (i.e. $\alpha' < 0$), neural net generalizes only if all paths are seen in training set, and test error drops exponentially in the scale of all paths are seen.

1 Introduction

In various tasks, neural networks trained on real data exhibit a power-law-like behavior concerning test error in relation to the number of training points [**bahri2021explaining**]. This behavior can be intuitively understood by considering that if a task is partitioned into numerous sub-tasks, each requiring a different scale of training points, it would result in a power-law-like pattern of test error versus training points.

Deriving an exact theoretical framework for real-world data, such as images or natural language, is challenging. It is difficult to ascertain precisely how tasks are divided into sub-tasks and to track the scale at which each sub-task is learned. Instead, researchers often resort to devising simple toy models of datasets where sub-tasks can be readily identified, from which they can deduce scaling exponents. Several prior works, such as [**hutter2021learning**] and [**michaud2024quantization**], have explored this approach. They introduced toy models for classification tasks, featuring numerous features with varying frequencies in the dataset. In these models, when a feature is observed, it is labeled correctly. These studies have demonstrated that when the frequency of features follows a Zipf law, i.e., the ith most common feature has a frequency on the order of $i^{-1-\alpha}$, the test error should follow $P^{-\frac{\alpha}{1+\alpha}}$, where P is the number of training points.

However, prior works have overlooked the hierarchical structure inherent in many real-world phenomena. For instance, most natural languages adhere to a context-free grammar, which follows a hierarchical tree structure. In this paper, we introduce the Random Hierarchy Model (RHM) [**cagnetta2024deep**], where each datum is generated from a hierarchical tree. We demonstrate that in this scenario, each sub-task is associated with identifying each generative path, not just the features in each layer. When distribution of feature in each layer follows zipf law with exponent α, distribution of generative path follows roughly zipf law with different exponent α', where we can derive α' from

38th Conference on Neural Information Processing Systems (NeurIPS 2024).

α. Furthermore, we show then when $\alpha > \alpha_c$ which means $\alpha' > 0$, we show that test error scales with exponent $\frac{\alpha'}{1+\alpha'}$.

2 Background

2.1 Test error when features are Zipf-distributed, Layer=1

This subsection follows [**hutter2021learning**]. Think about there are some class labels, and for each class label we assign number of features. Note that we are not considering hierarchical structure(or we could think of it as layer=1). Let's assume that's once the feature is seen in training set, neural net gets the label correctly. Thus initial task is divided into many different sub-tasks, which is memorizing label given for each feature. Rank the frequency of features, and denote ith biggest frequency as f_i. Note that $\sum_i f_i = 1$. The probability to not see ith feature in P training samples is $(1-f_i)^P$. Each sub-task will contribute to test error with factor f_i. Thus test error is $\sum_i f_i(1-f_i)^P$. When $Pf \ll 1$, this could be approximated to superposition of exponential.

$$\epsilon = \sum_i f_i(1-f_i)^P \sim \sum_i f_i e^{-P_i} \tag{1}$$

Where ϵ stands for test error(1-accuracy). This means that test error can be thought of as superposition of exponential with different characteristic scale.

We can approximate this sum by integral,

$$\epsilon = \sum_{i=1}^{\infty} f(i)(1-f(i))^P \sim \int_1^{\infty} dx f(x) e^{-Pf(x)} \tag{2}$$

$$= \int_0^{f_1} \frac{du\, u e^{-Pu}}{|f'(f^{-1}(u))|} \text{ (where } u = f(x)) \sim \frac{1}{P^2 |f'(f^{-1}(\frac{1}{P}))|} \tag{3}$$

Last approximation is because numerator ue^{-Pu} is maximal and strongly concentrated at $\frac{1}{P}$.

Now let's see the case where frequency follows zipf-law, as in natural languages. $f_i \sim i^{-(\alpha+1)}$ for some $\alpha > 0$. Then $u = f(x) = \alpha x^{-(\alpha+1)}$. This implies $x = f^{-1}(u) = (u/\alpha)^{-\frac{1}{1+\alpha}}$ and $f'(x) = -\alpha(\alpha+1)x^{-(\alpha+2)} = -\alpha(\alpha+1)(u/\alpha)^{(\alpha+2)/(\alpha+1)}$, hence

$$\epsilon \sim \frac{1}{P^2 |f'(f^{-1}(\frac{1}{P}))|} \sim P^{-\frac{\alpha}{1+\alpha}} \tag{4}$$

That is, Zipf-distributed data (with exponent $\alpha+1$) lead to a power-law learning curve (with exponent $\frac{\alpha}{1+\alpha}$). Caveat here is that this power-law will hold only if there are infinitely many feature. When there are only finite number of feature, test error would eventually be dominated by exponential of smallest frequency, $e^{-Pf_{min}}$.

2.2 Toy model of hierarchical data: Random Hierarchy Model

The assumption in above subsection that data is generated by one layer(i.e. each feature is directly associated to class label) is unrealistic for real data. Language, Image data should have been produced by complex, hierarchical process. Although it would be extremely difficult to mimic real dataset, let's implement the toy model of dataset that is generated from hiearchy, but has simplest form, which is tree(also known as context free grammer). In Random Hierarchy Model[**cagnetta2024deep**], each input is generated from tree structure all the way from class label. Each n_c class is mapped into m lower level synoynyms(s -tuple made of vocabulary size v)m which we call *rules*. For subsequent layers, each v higher level feature is mapped into m lower level synoynyms(s -tuple made of vocabulary size v). To generate dataset, we have to start from class label and go down hierarchy, *sampling* among m synonyms. Note that in the original model, sampling among m synonyms was uniform.

In the work of [**cagnetta2024deep**], when each feature is seen uniformly, there is one characteristic scale of training point where neural net learns classification task. This was deduced by considering

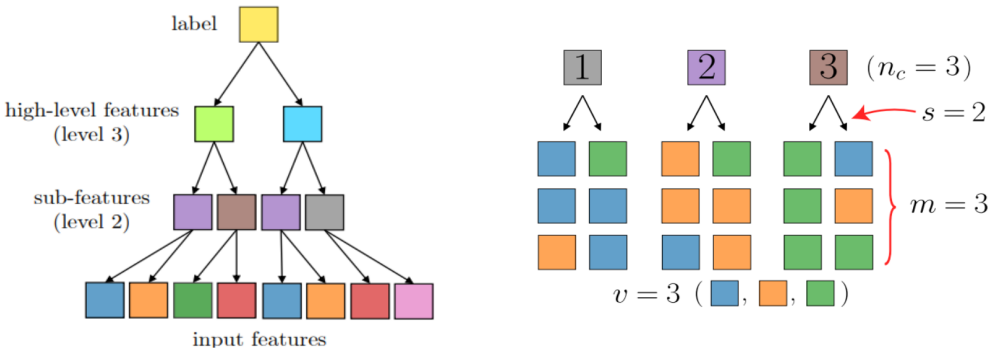

Figure 1: **Left:** Hierarchy generating input data in RHM. **Left:** How higher level feature is mapped into lower level synonyms. We use notation such that n_c is number of classes, s is tuple size, L is number of layers, v is number of feature, m is number of synonyms

local correlation of last layer(bottom-most) layer with class label. Due to the randomness of assigning *rules*, there occurs correlation between feature and sub-feature, which eventually lead to correlation between class label and last layer tuple. Note that last layer tuple that is mapped by same higher level feature would have same strength of correlation. The characteristic scale is then simply the scale when sampling noise gets smaller than true correlation, so that neural net can group different last layer tuples to same group, learning *invariance* between change of synonyms(i.e. changing lower level tuple which are mapped by same higher level feature). The fact that neural net learns the task by such process could be checked by applying homogeneous *rules* so that there is no correlation, then generalization is not achieved in this scale.

To be more specific, given a class α, the tuple $\boldsymbol{\mu}$ appearing in the j-th input patch of last layer, is determined by a sequence of L choices—one choice per level of the hierarchy—of one among m possible lower-level representations. These m^L possibilities lead to all the mv distinct input s-tuples. Occurrence of tuple μ given class α (denote this as $N_j(\boldsymbol{\mu}; \alpha)$) happens $m^L/(mv)$ times on average. Now let's see the fluctuation of conditional probabilities $\frac{N_j(\boldsymbol{\mu};\alpha)}{N_j(\boldsymbol{\mu})}$. If n_c is relatively large, fluctuation of this probabilities are dominated by fluctuation of $N_j(\boldsymbol{\mu}; \alpha)$. Under the assumption of independence of the m^L choices, the fluctuations of $N_j(\boldsymbol{\mu}; \alpha)$ relative to its mean are given by the central limit theorem and read $(m^L/(mv))^{-1/2}$ in the limit of large m. Therefore, the relative fluctuations of $\frac{N_j(\boldsymbol{\mu};\alpha)}{N_j(\boldsymbol{\mu})}$is order of $(m^L/(mv))^{-1/2}$. We call this *signal*.

The magnitude of the *noise* is given by the ratio between the standard deviation and mean, over independent samplings of a training set of fixed size P, of the empirical conditional probabilities $\frac{\hat{N}_j(\boldsymbol{\mu};\alpha)}{\hat{N}_j(\boldsymbol{\mu})}$. Only $P/(n_c mv)$ of the training points will on average, belong to class α while displaying feature μ in the j-th patch. Therefore, by the convergence of the empirical measure to the true probability, the sampling fluctuations relative to the mean are of order $[P/(n_c mv)]^{-1/2}$. Balancing signal and noise yields the characteristic $P^\star$ for the emergence of correlations.

$$P^\star \sim n_c m^L \tag{5}$$

Note that if we think of generative path from class label to last layer tuple in fixed position, freuqency of specific path is $\frac{1}{n_c m^L}$, Thus $P^\star$ could be interpreted as scale at which given path is seen once. Furthermore, since every path is uniform, scale that most paths are seen should be order of $P^\star$. We can check test error trained on neural net in this case, and we can observe that test error drops abruptly(exponential behavior) at training point $P^\star$. For CNNs, the actual scale when test error drops was indeed order of $P^\star$. For FCNs, we need $2^L P^\star$ training points.

3 Test error result

The setting we used is $m = v = n_c = 10, s = 2, L = 3$. For vanilla RHM, $P^\star = 10^4$. Here we had many different schemes of choosing frequency of features, some layer we put zipf law with exponent $-(1+\alpha)$, where we left other layer uniform.

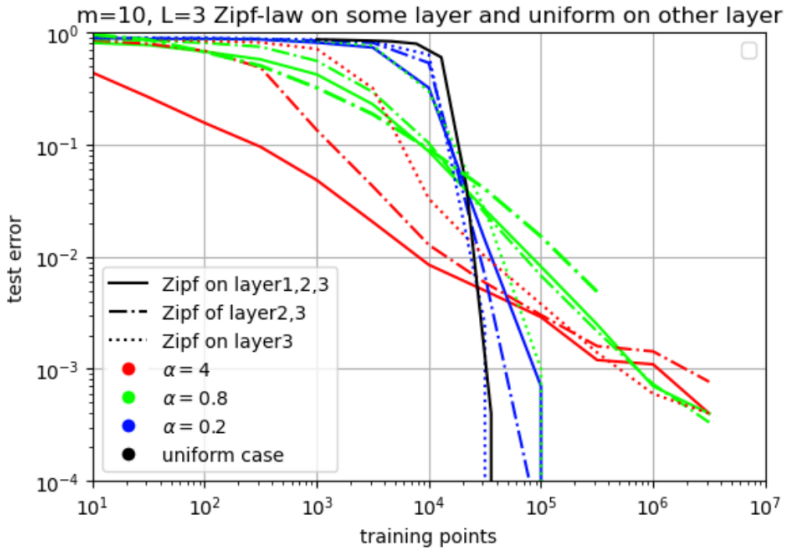

Figure 2: Test error for observed. Dotted line is test error achieve when nonuniformly sampling only for layer 3, and still uniformly sampling in other layer 1,2. Dashdotted line is when nonuniformly sampling for layer 2,3, and still uniformly sampling in layer 1. Solid lines for nonuniformly sampling for layer 1,2,3. Note that by implementing nonuniform sampling where frequency of each layer is distributed by power-law, we attain power-law like test error curve, not exponential one as vanila RHM. As we implement nonuniform sampling to more and more layers, the starting point where test error drop decreases. For instance, when we implement nonuniform sampling only to last layer, test error drops from $P \sim 10^3$. If we implement nonuniform sampling to last two layers, test error drops from $P \sim 10^2$. If we implement nonuniform sampling to all 3 layers, test error drops from $P \sim 10$.

4 Path as natural subtask of hiearchy

When L=1, as we showed above, subtask is directly related to each feature, which means production rule from class label to s-tuple. However, when $L > 1$, things are not that simple. In this case, there are hidden layers between class label and input patch. For instance, consider L=2 case. Think about the case where class label a is mapped to (b, c) layer1 tuple by production rule. Then layer 1 feature b is mapped to (d, e) by production rule. In this case, we can not know immediately that (d, e) came from L=1 feature b, since only thing given is last layer and class label. The most natural unit is then each generative path, not just production rule in each layer.

This also explains the starting point where test error starts to drop. In our model, each geneative path has its own frequency. Then we would think that each path has characteristic scale of training point that it has seen once, denote this $P^\star_{f_1 f_2 f_3} = \frac{1}{f_1 f_2 f_3}$, where f_l is frequency of choice that path taken at $L = l$.

In the test error curve for $m = v = n_c = 10, L = 3, s = 2$ when we implement nonuniform sampling only to last layer, test error drops from $P \sim 10^3$. If we implement nonuniform sampling to last two layers, test error drops from $P \sim 10^2$. If we implement nonuniform sampling to all 3 layers, test

error drops from $P \sim 10$. This is directly captured by characteristic scale of path.

$$\begin{aligned}
\text{uniform case: } P^\star_{f_1 f_2 f_3} &= P^\star_{\frac{1}{m}\frac{1}{m}\frac{1}{m}} = 10^4 \\
\text{nonuniform in one layer: } P^\star_{f_1 f_2 f_3} &= P^\star_{f_1 \frac{1}{m}\frac{1}{m}} \in [10^3, 10^4] \\
\text{nonuniform in two layers: } P^\star_{f_1 f_2 f_3} &= P^\star_{f_1 f_2 \frac{1}{m}} \in [10^2, 10^4] \\
\text{nonuniform in three layers: } P^\star_{f_1 f_2 f_3} &\in [10^1, 10^4]
\end{aligned} \tag{6}$$

5 Characteristic scale of α from parameter of dataset(idea of israeli postoc we met last week)

When path is indeed what governs learning, we could deduce one natural charaterstic scale of α from parameter of dataset(m, L). When we implement zipf law on every layer with exponent $-(1+\alpha)$, the most frequent path will be scale with 1^L, and there is only one of these since we have to choose biggest one in every layer. Denote number of such path N_1. Second frequent path will be scale with $1^{L-1}2^{-(1+\alpha)}$, and there L possible ways to have such path, $N_2 = L$. Third frequent path will be scale with $1^{L-1}3^{-(1+\alpha)}$, and there L possible ways to have such path, $N_3 = L$. Fourth frequent path will be scale with $1^{L-1}4^{-(1+\alpha)} = 1^{L-2}2^{-2(1+\alpha)}$, and $N_4 = L + \binom{L}{2}$. Then eventually frequency of path will approximately follow exponent zipf law with exponent $-(1+\alpha')$, and $(\sum_{i=1}^{j} N_i)^{-(1+\alpha')} \sim j^{-(1+\alpha)}$.

This becomes a number theory problem, and threre should be known result with approximation. The Characteristic scale for α in this case is when effect of dropping freuqency of path is comparable to effect of increment to $\sum_{i=1}^{j} N_i$, so that $\alpha' = 0$. Denote this α_c.

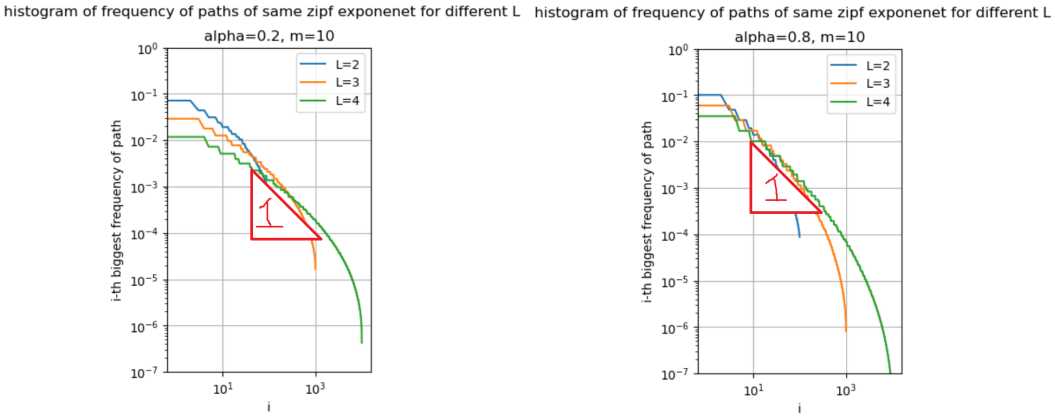

Figure 3: Zipf plot of path when we implement zipf law with exponent $-(1+\alpha)$ in all layers, and we change number of total layer. As we increase total number of layer, α' decreases, and eventaully be smaller than 0. When $m = 10, L = 3$, $\alpha' \sim 0.2$. and when $m = 10, L = 4$, $\alpha' \sim 0.8$.

6 Scaling description of test error when $\alpha > \alpha_c$

Accuracy as how many path seen in training set: For each datum, there is s^{L-1} number of paths, each leading to different position of last layer tuple all the way from class label. We splitted test set into how many paths were seen in training set, and plot accuracy as training point averaging ten trials.

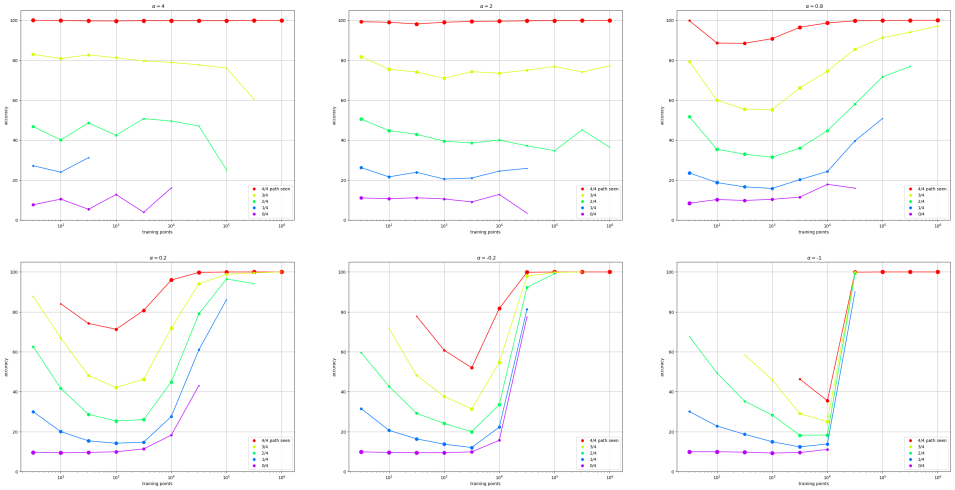

Figure 4: Accuracy of splited test set into how many paths are seen in training set. $m = v = n_c = 10, L = 3, s = 2$. $P^\star = 10^4$. Marker size correpsonds to logarithm of splitted test size. In this case, numerically $\alpha_c \sim 0.2$, and for $\alpha > \alpha_c$, accuracy of splitted test set was relatively flat line. Moreover as expected, when more path are seen in training set accuracy gets higher.$\alpha > \alpha_c$ however it seems more like there is sudden scale that accuracy increases for all splited test set, which is around $P^\star$ from uniform case(i.e. $\alpha = -1$ in all layer).

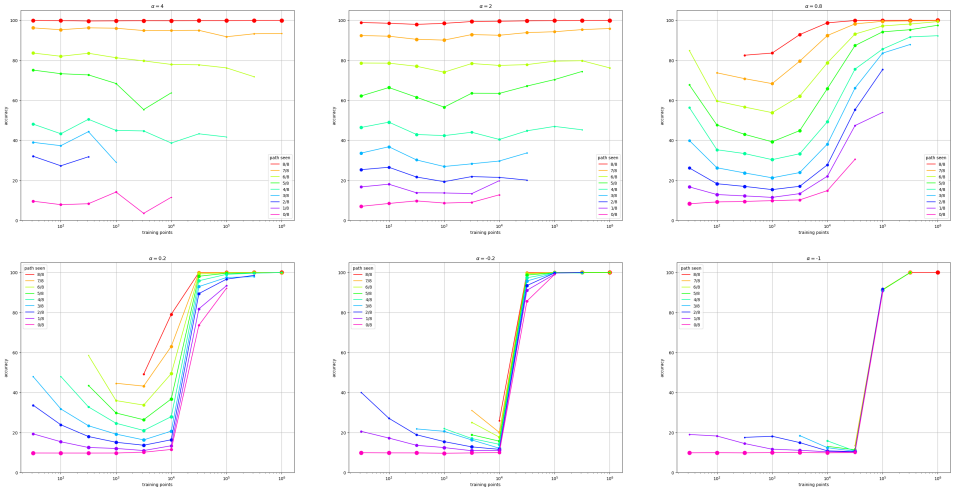

Figure 5: Accuracy of splited test set into how many paths are seen in training set. $m = v = n_c = 10, L = 4, s = 2$. $P^\star = 10^5$. Marker size correpsonds to logarithm of splitted test size. In this case, numerically $\alpha_c \sim 0.8$, and for $\alpha > \alpha_c$, accuracy of splitted test set was relatively flat line. Moreover as expected, when more path are seen in training set accuracy gets higher.$\alpha > \alpha_c$ however it seems more like there is sudden scale that accuracy increases for all splited test set, which is around $P^\star$ from uniform case(i.e. $\alpha = -1$ in all layer).

Scaling exponent of test error when $\alpha > \alpha_c$:

For fixed position of input(last layer), let's denote $path(P)$ the probability that this tuple is generated by path that has $P^\star_{f^{L=1},f^{L=2},f^{L=3}} \leq P$

$$path(P) = \sum_{i_1}\sum_{i_2}\sum_{i_3} f_{i_1}^{L=1} f_{i_2}^{L=2} f_{i_3}^{L=3} \text{ s.t. } P^\star_{f^2 f^2 f^3 < P} \tag{7}$$

$$\sim 1 - cP^{-\frac{\alpha'}{1+\alpha'}} \text{ for some } c \tag{8}$$

Assume path is independent, and also assume accuracy was flat line as function of training point for each splitted test set(For $\alpha > \alpha_c$ this is roughly true). Denote c4 is accuracy when all four paths are seen, c3 is accuracy when three out of four path is seen etc.

$$\epsilon = 1 - c_4\binom{4}{4}(path(P))^4 - c_3\binom{4}{3}(path(P))^3 - c_2\binom{4}{2}(path(P))^2 - c_1\binom{4}{1}(path(P))^1 - c_0\binom{4}{0}(path(P))^0 \tag{9}$$

$$\sim \text{some fourth order polynomial function of } P^{-\frac{\alpha'}{1+\alpha'}} \tag{10}$$

$$\sim P^{-\frac{\alpha'}{1+\alpha'}} \tag{11}$$

Result on RHM:

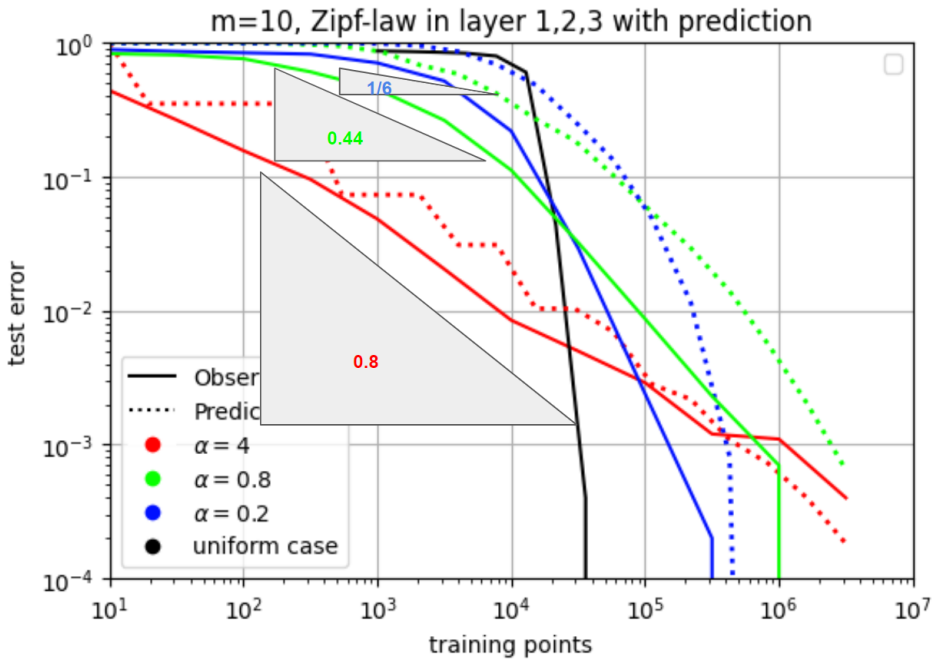

Figure 6: We used setting $m = n_c = v = 10, L = 3, s = 2$. Here we are nonuniformly sampled all three layers with frequency following zipf law with exponent $1 + \alpha$. We plotted uniform case for comparison. Not that since we only have $m = 10$, test error is not true power-law, and drops exponentially eventually. We also draw traingle correspoding to slope of theoretical prediction $\frac{\alpha'}{1+\alpha'}$. Forget the number in triangle it's $\frac{\alpha}{1+\alpha}$ I will fix it.

7 Possible algorithm to explain such behavior:

We hypothesize here that in order to learn RHM, not only each path should have seen in training set, but also neural net should be able to group representation of each path. When $\alpha >> \alpha_c$, simply seeing path will be able to group representaion as we will show below. However in the opposite limit, when $\alpha = -1$ where all synoynyms are uniform, then neural net will only be able to group representaion as long as it sees most of its path so that by local correlation of class label and input patch it can group representation as shown in section 2.

Limit $alpha = -1$**:** Local correlation of class label and input patch is only way to group representations. In this case accuracy for each splitted test set abruptly increases when most of the paths are seen, which happens when training point is order $P^\star$

Limit $alpha >> \alpha_c$**:** Think about the limit when α is large. Then one synonym will dominate m synonym, and each position in position of input can be associated to class label directly. The next possibility is taking next frequent choice in just one place in whole hierchical tree. Think about the case where this choice was in last layer. Then, neural net could directly group this newly chosen tuple to most frequent tuple given classl label. This could be justfied by simply training on patch wise perceptron that we train linear layer for each patch and add each logits to get total logit. Indeed, when α is big, patchwise perceptron shows similar accuray for splitted test set.

intermediate α**:** Something middle between these two limits.

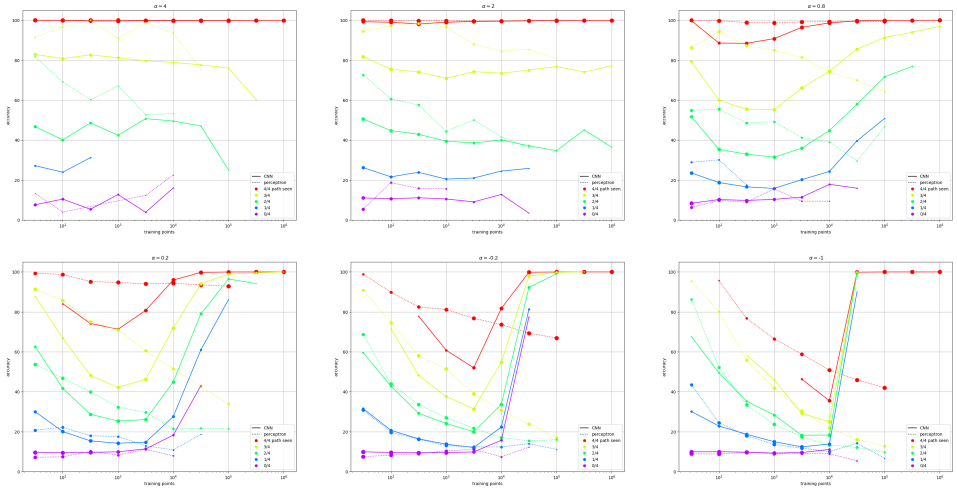

Figure 7: Accuracy of splited test set into how many paths are seen in training set. $m = v = n_c = 10, L = 3, s = 2$. $P^\star = 10^4$. Solid line corresponds to 3 layer CNN, while dotted line correspond to patchwise perceptron. Marker size correpsonds to logarithm of splitted test size. In this case, numerically $\alpha_c \sim 0.2$, and for $\alpha > \alpha_c$, accuracy of splitted test set was relatively flat line. For $\alpha = 4, 2$ patchwise perceptron shows similar accraucy for spllited test set.

8 Discussion

Appendix

A Other plots

Splitted test according to path, varying v:

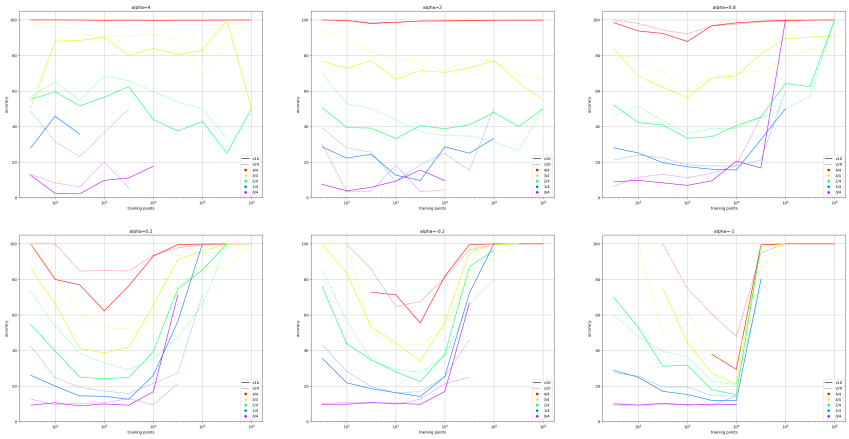

Figure 8: Accuracy of splited test set into how many paths are seen in training set. $m = n_c = 10, L = 3, s = 2$. $P^\star = 10^4$. We vary $v = 10, 20$. We can see that since definition is choices of synonym not feature, varying v dose not have effect of accuracy of splitted test set.

Splitted test according to path, Zipf law on just one layer and uniform on other two:

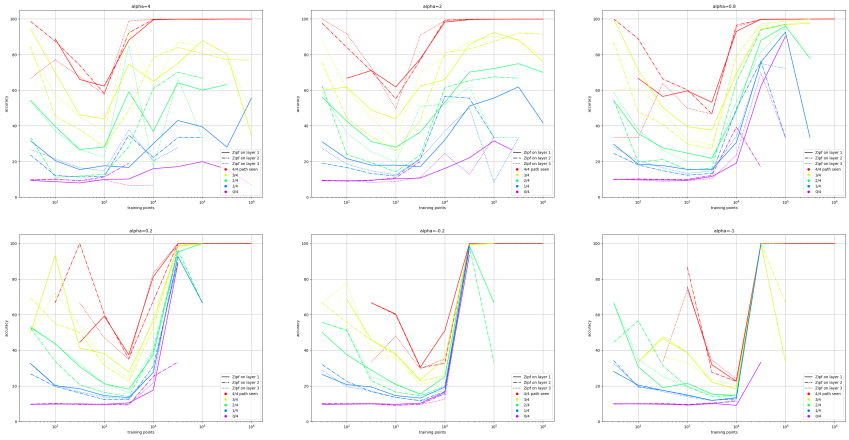

Figure 9: Accuracy of splited test set into how many paths are seen in training set. $m = v = n_c = 10, L = 3, s = 2$. $P^\star = 10^4$. Solid line corresponds implementing zipf law on top layer and uniform on othes, Dashdotted line corresponds implmenting zipf law on middle layer and uniform on others, and Dotted line corresponds implementing zipf law of bottom layer and uniform on others.No surprsingly, accuracy of splitted test set shows similar tendency whether we implemnt zipf law on which layer, showing that path is indeed what we should look.