Laboratory for Information and Inference Systems
EPFL STI IEL LIONS
ELE 233 (Bâtiment ELE)
Station 11
CH-1015 Lausanne

Master Semester Project

Dynamic Polynomial Neural Networks

Alec Flowers

Course of Study:	Data Science
Examiner:	Prof. Dr. Volkan Cevher
Supervisor:	Dr. Grigorios Chrysos
Commenced:	February 21, 2022
Completed:	June 10, 2022

1 Introduction

Polynomial neural networks are neural network architectures that can be provably shown to model a high-dimensional polynomial of the input. They inherently exploit multiplicative interactions between the input data points and learn how to weight these interactions to best accomplish the stated task. They can learn without non-linear activations, and with non-linear activations can produce state-of-the-art results in domains such as image generation and facial identification. Chrysos et al.

Many current successful network architectures fit within the polynomial framework. A ResNet He et al. [2015] is a 1st degree polynomial, a squeeze and excitation network Hu et al. is a second degree polynomial, a non-local neural network Wang et al. [b] is a third degree polynomial, and extensions to n-degree polynomials were introduced as Π-nets Chrysos et al.. By leveraging the polynomial framework, base models can be transformed into polynomials to create networks that are more expressive, have fewer parameters, and can surpass the accuracy of the base model Chrysos et al. [2021].

In this paper, we also wanted to leverage the polynomial framework to build, in a principled way, the best networks possible. The best network could be measured in many way, and we are focused on improving accuracy by increasing the networks representation power while still generalizing well and not overfitting.

An important question to address when using a polynomial network is, what is the best polynomial degree given our dataset?

A traditional approach would be to view the degree as a hyperparameter that could be tuned and to find an optimal value using grid search, neural architecture search, or a meta-learning approach. We are interested in taking a less traditional view and attempt to find the optimal polynomial degree per individual sample in a dataset. This means having a dynamic architecture that can adapt conditioned on the input.

Dynamic architectures per sample are often developed with one of two goals in mind. The first is to increase efficiency by changing the architecture to allocate appropriate computation per sample. The second is to increase representation power by adapting the networks parameters Han et al.. We are interested in increasing representation power.

Since we are interested in increasing representation power of the network, before we actually build such a dynamic network we need to validate the hypothesis that:

Dynamically adjusting the polynomial expansion per data sample will increase the representation power of polynomial networks.

We had four reasons why we believe this hypothesis was true and worth investigating:

1. Often the top 1 accuracy is reported in papers, but the per-class accuracy can often vary significantly. There is room to improve on per-class accuracy which we believe is a place where a dynamic network can shine.
2. The transformer architecture Vaswani et al. [2017] which has caused a revolution in deep learning has a soft weighting mechanism that works as a sort of memory system and is dynamic conditioned on the input. This dynamic scheme highlights the usefulness in adapting a network to every sample in the dataset.
3. We can look at a dynamic network from an ensemble view and it is the same as selecting the best expert from a mixture of experts per sample. In this case, the mixture of experts (MoE) to be selected from are the different degrees of polynomials. This MoE approach has been used to increase model capacity without a proportional increase in computation Shazeer et al..
4. A dynamic variation of the degree would best allocate both the resources (compute) as well as align the expressivity and strengths of a certain degree polynomial to a sample.

Taking a step back even further, there are problems with the current trend of training larger and larger models that range from environmental concerns to loss in accessibility due to the engineering complexity and compute resources Bommasani et al. [2021]. We believe that dynamic architectures

will play an increasingly important role in the field of deep learning as they become expressive alternatives to increasing model sizes into the trillions of parameters.

2 Polynomial Neural Networks

Polynomial neural networks form the foundation for our research question. We provide a brief overview and explanation of what a polynomial network is along with two decomposition's that provide us a recursive formulation which we use to build neural networks to run our experiments.

Notation: $\times_m$ is the mode m vector product, $*$ is the hadamard product, and $\odot$ is the Khatari-Rao product.

A polynomial neural network has a specific structure that is mathematically equivalent to a tensor decomposition of a high dimensional polynomial. This means that a network with this structure is learning weights for a high dimensional polynomial in order to best approximate the unknown distribution. We can formulate a high dimensional polynomial of an input $\mathbf{z} \in \mathbb{R}^d$ as a function $G : \mathbb{R}^d \to \mathbb{R}^o$

$$\mathbf{x} = G(\mathbf{z}) = \sum_{n=1}^{N} (\mathbf{W}^{[n]} \prod_{j=1}^{n+1} \times_j \mathbf{z}) + \beta \tag{1}$$

with $\beta \in \mathbb{R}^d$ and $\{\mathbf{W}^{[n]} \in \mathbb{R}^{o \times \prod_{m=1}^{n} \times_m d}\}_{n=1}^{N}$ as the learnable parameters. The polynomial in equation 1 can be reformulated as a tensor decomposition and written as a recursive formula. Using a tensor decomposition is a key insight that ensures the number of parameters doesn't explode exponentially with the polynomial degree. Using a coupled CP tensor decomposition (CCP) we get the following recursive relationship:

$$\mathbf{x}_n = (\mathbf{U}_{[n]}^T \mathbf{z}) * \mathbf{x}_{n-1} + \mathbf{x}_{n-1} \tag{2}$$

for $n = 2, ..., N$ with $\mathbf{x}_1 = \mathbf{U}_{[1]}^T \mathbf{z}$ and $\mathbf{x} = \mathbf{C}\mathbf{x}_N + \beta$. The relationship between the recursive formulation and the high dimensional polynomial can be made explicit with an example for $N = 3$

$$G(\mathbf{z}) = \beta + \mathbf{W}_{[1]}\mathbf{z} + \mathbf{W}_{[2]}(\mathbf{z} \odot \mathbf{z}) + \mathbf{W}_{[3]}(\mathbf{z} \odot \mathbf{z} \odot \mathbf{z})$$

where we write out the CCP decomposition for each weight matrix:

- $\mathbf{W}_{[1]} = \mathbf{C}\mathbf{U}_{[1]}^T$
- $\mathbf{W}_{[2]} = \mathbf{C}(\mathbf{U}_{[3]} \odot \mathbf{U}_{[1]})^T + \mathbf{C}(\mathbf{U}_{[2]} \odot \mathbf{U}_{[1]})^T$
- $\mathbf{W}_{[3]} = \mathbf{C}(\mathbf{U}_{[3]} \odot \mathbf{U}_{[2]} \odot \mathbf{U}_{[1]})^T$

Using a nested coupled CP decomposition (NCP) gives us the following recursion:

$$\mathbf{x}_n = (\mathbf{A}_{[n]}^T \mathbf{z}) * (\mathbf{S}_{[n]}^T \mathbf{x}_{n-1} + \mathbf{B}_{[n]}^T \mathbf{b}_{[n]}) \tag{3}$$

With this formulation it is easy to modify any neural network to turn it into a polynomial neural network. This is because the second term in equation 3 is the normal formulation for a fully connected layer with weights $\mathbf{S}_{[n]}^T$ and bias $\mathbf{B}_{[n]}^T \mathbf{b}_{[n]}$. Turning the normal formulation into this is as simple as adding a skip connection and applying the hadamard product.

3 Initial Experiments

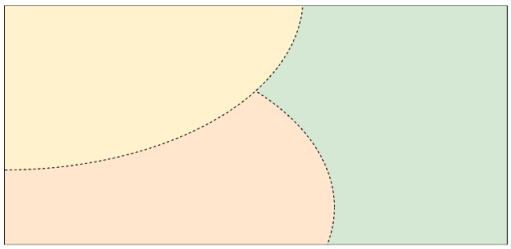


Figure 1: Visualization of partitioning the latent space

Our first task was to validate our hypothesis. Our hypothesis relies on the fact that adjusting the degree of the polynomial changes the expressivity of the network in some way. If this doesn't occur, we would be selecting between two networks that had learned a very similar function of the data and there would be no point in making a decision between the two of them. In order to validate our hypothesis we need to show that a low degree and a high degree polynomial learn a different function of the data.

3.1 FashionMNIST

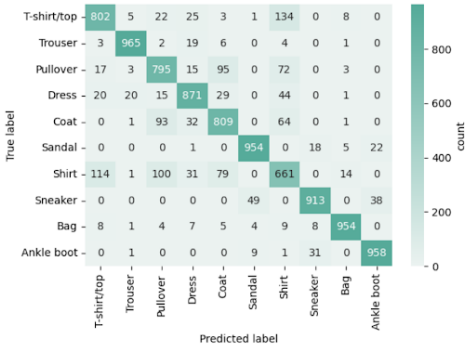
The first experiment was to run a 4 degree CCP polynomial (CCP-low) and a 16 degree CCP polynomial (CCP-high) on FashionMNIST and visualize the confusion matrix, accuracy per class, and display TSNE plots after different layers. To understand the behavior of vanilla polynomial networks and to eliminate various confounding factors, no activation functions nor regularization such as L2 penalty, dropout, or batchnorm were used in these initial experiments.

	degree	parameters	hyperparams	optimizer
CCP-low	4	50378	epochs: 50 hidden: 16 batch size: 128	SGD momentum: 0.9 lr: 0.001
CCP-high	16	200906	epochs: 50 hidden: 16 batch size: 128	SGD momentum: 0.9 lr: 0.001

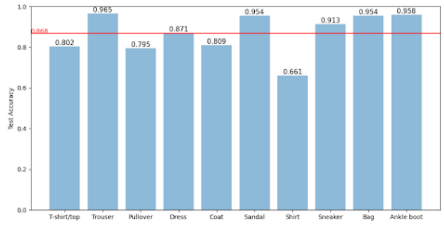
Some key highlights from this experiment:

- While there is some variance in the per class accuracy, these networks seemed to learn a very similar function of the data. They both do well and poorly on the same classes.
- In terms of overall accuracy, they do very similar. This was surprising as CCP-high has 4x the number of parameters and my expectation is that it would do better than the CCP-low, especially on "difficult" classes.
- We now have an idea of which classes are difficult for this dataset. The networks really struggles with the Shirt category and confuses it with Pullover, Coat, and T-shirt/top.
- The TSNE plots seem to show that especially on CCP-high, the later layers are not that helpful in separating out classes. Even as we move along to later layers the difficult classes stay mixed and the easy classes stay clustered. This seems to suggest that the higher degree is not able to provide more expressive power.

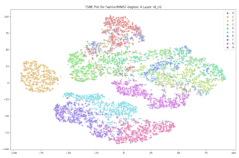
In the Meta-Neighborhoods Shan et al. paper they split up a latent representation of the input space and this gave me some initial ideas and guided my thinking during the initial experimentation phase. Meta-Neighborhoods is a technique that maps a sample into a neighborhood and then adapts the model parameters to the neighborhood of the input. The neighborhoods and the model are learned together using a meta-learning training technique. They show that these neighbors represent informative categories when such information is not explicitly given during training. This is a practical implementation of a dynamic network that changes the entire network conditioned on the input itself.



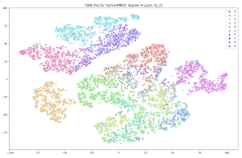
(a) Confusion Matrix



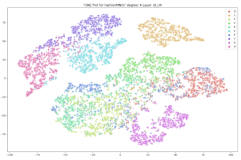
(b) Per Class accuracy



(c) Layer 2 TSNE Plot

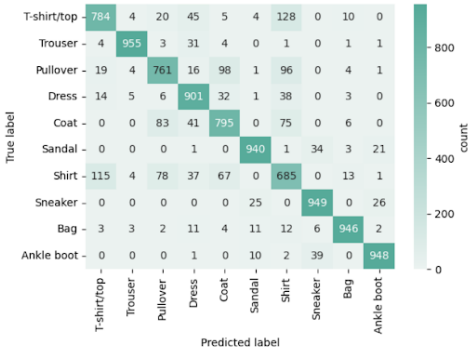


(d) Layer 3 TSNE Plot

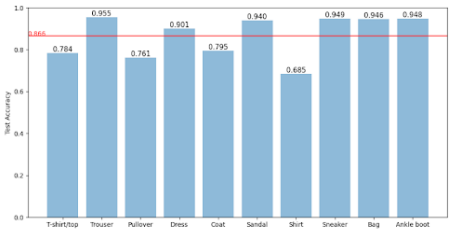


(e) Layer 4 TSNE Plot

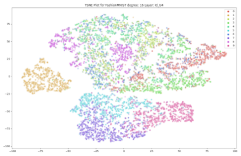
Figure 2: CCP-low tested on FashionMNIST



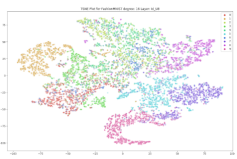
(a) Confusion Matrix



(b) Per Class accuracy

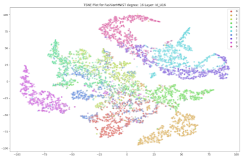


(c) Layer 4 TSNE Plot



(d) Layer 8 TSNE Plot

(e) Layer 12 TSNE Plot



(f) Layer 16 TSNE Plot

Figure 3: CCP-high tested on FashionMNIST

Intuition says that "easy" data samples should need less strength and therefore a low degree polynomial, while "difficult" classes should get more strength and therefore a high degree polynomial. What is considered "easy" and "hard" could be learned and drawing on the Meta-Neighborhoods paper, we can imagine learning a dictionary which partitions the latent space and assigns each neighborhood the appropriate polynomial degree.

We tried multiple experiments with both FashionMNIST and CIFAR10 where we manually split up the input space. For example, turning the problem into a binary classification problem between T-shirt and all others and seeing if a low or high degree could be more successful in this case. However, each of these experiments were not successful and did not show a concrete difference between the different polynomial degrees. We thought this naive splitting up of the input space by class likely

did not align with the strengths and weaknesses of the network itself, but rather how we wanted the networks to function.

3.2 Spurious Correlation

So far we had seen no evidence that a high degree polynomial was better than a low degree polynomial even though it has more representation power because it has a larger number of layers and parameters. We wanted to design an experiment to find a case where a high degree would outperform a low degree.

While over-parameterized networks have the capacity to memorize random data Zhang et al. [2016], recent work has shown that they still prioritize learning simple patterns first Arpit et al. [2017] and experimentally they exhibit good generalization behavior. We try to exploit this behavior to design an experiment where an over-parameterized network will perform better than a smaller network.

This experiment was based around the idea of introducing a spurious correlation of color to the MNIST dataset during training. In testing, we take away this spurious correlation and color the digits all one color to see which network relied on this correlation the most. The spurious correlation we introduced was numbers [0, 1, 2, 3, 4] would have a color distribution of 70% red, 20% green, and 10% blue while numbers [5, 6, 7, 8, 9] would have a color distribution of 70% blue, 20% red, and 10% green.

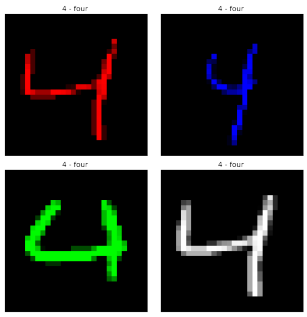


Figure 4: Example of 4's of different colors.

The hypothesis was that a low degree polynomial would rely on the spurious correlation while a higher degree would learn the correlation but still learn a simple function and be able to ignore the spurious correlation.

After training on the spurious correlated data, we test on homogeneous data that is all one color. Uniform corresponds to a white number where all the channels are populated with data. All Red, All Green and All Blue, correspond to only 1 channel being populated with data (r, g, b) while the other channels are filled with 0's.

Table 1: Test Accuracy

	2	4	8	16
Uniform	**.898**	.870	.857	.731
All Red	.900	**.953**	.947	.885
All Green	.900	**.943**	.931	.849
All Blue	.901	**.938**	.937	.873

Table 2: Digits 0-4 Test Accuracy

	2	4	8	16
Uniform	**.9262**	.9010	.8316	.7934
All Red	.9656	.9698	**.9794**	.9534
All Green	.9528	.9612	**.9706**	.9158
All Blue	.8748	.9104	**.9144**	.8140

Table 3: Digits 5-9 Test Accuracy

	2	4	8	16
Uniform	.8654	.8320	**.8730**	.6618
All Red	.8296	**.9346**	.9112	.8130
All Green	.8428	**.9220**	.8888	.7762
All Blue	.9258	**.9658**	.9584	.9324

At the outset it seems as if the 4th degree is the best network, but when we split apart the accuracy into buckets the picture becomes more complex. The 8th degree polynomial has a slight edge on digits [0-4] while the 4th degree polynomial holds the edge on digits [5-9]. Our hypothesis that a low degree polynomial will capture the noise does seem to be the case with the 2nd degree as we see it has a much lower accuracy on the test data when the color comes from a less sampled class (all blue in digits [0-4] and all red in digits [5-9]). The 16 degree polynomial didn't train very stably and therefore its accuracy is much below the other networks.

We can consider the Uniform case to be out-of-distribution examples as the nets were never shown these types of digits during training. In this case the lower degree seems to be able to generalize better and be more robust than the higher degrees.

This is the first experiment where we see some small differences in learning between lower and higher degree polynomials.

3.3 Spectral Filtering

Recent work highlights that the family of Π-nets speeds up the learning of higher frequencies, and this is postulated to be the reason why they are effective at image generation and facial recognition where high frequency information is crucial Choraria et al.. Extending this thinking further we wanted to understand if increasing the degree of the polynomial also speeds up the learning of higher frequencies. If so, this difference in learning, in expressivity, between a lower and higher degree polynomial could help us validate our hypothesis. A low degree polynomial would be better employed in classification when lower frequency information is available or low frequency information is more indicative of the correct class, and a high degree polynomial would be better employed when higher frequency information is available or high frequency information is more indicative of the correct class.

To design an experiment we take inspiration from Wang et al. [a] where they filter images for frequency content based on a frequency radius and inspect prediction accuracy when using heuristics such as Dropout, Mixup, and BatchNorm to see how they react to low and high frequency components.

In our experiment we filter images for different levels of high and low frequency content as can be seen in 5.

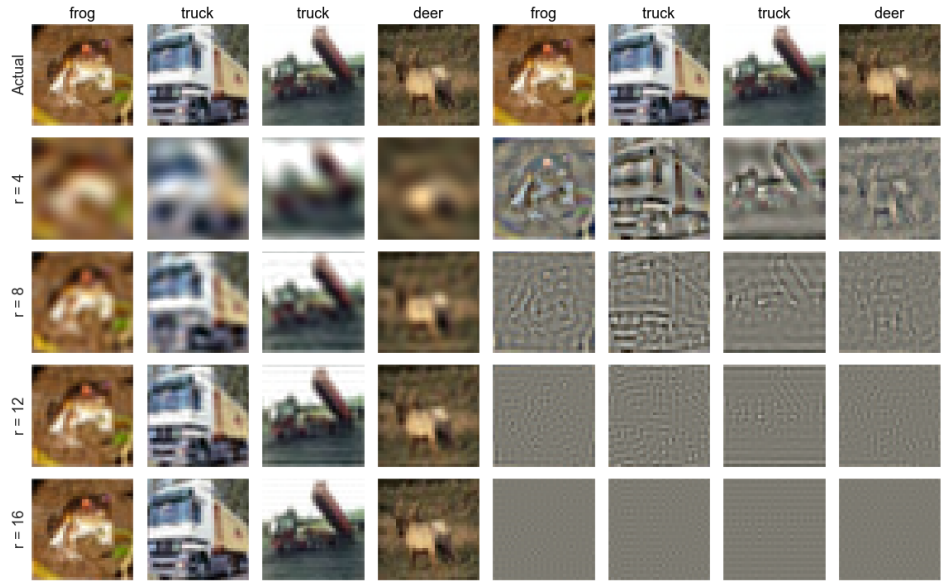


Figure 5: The first 4 columns show 4 samples filtered for low frequency. The last 4 columns show 4 samples filtered for high frequency content. The smaller the radius the more high frequency content and vice versa. These samples are what the polynomial nets were trained on.

We compare two different architectures a 2nd degree CCP polynomial (CCP-low) and a 6th degree CCP polynomial (CCP-high). The expectation was that a high degree polynomial would be able to learn more from the high frequency information, while a low degree polynomial would learn more from the low frequency information. We conducted 9 training runs each for CCP-low and CCP-high, 4 runs trained on low frequency information with radius [4, 8, 12, 16], 4 runs trained on high frequency information with radius [4, 8, 12, 16], and 1 baseline run on actual CIFAR10 images.

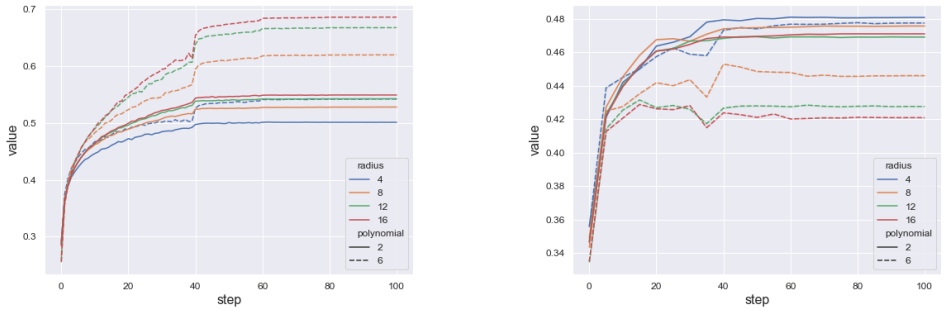
	degree	parameters	hyperparams	optimizer
CCP-low	2	98506	epochs: 100 hidden: 16 batch size: 100	SGD momentum: 0.9 lr: 0.0001 lr step: [40, 60 , 80, 100] gamma: 0.1
CCP-high	6	295114	epochs: 100 hidden: 16 batch size: 100	SGD momentum: 0.9 lr: 0.0001 lr step: [40, 60 , 80, 100] gamma: 0.1

We visualize the training and validation accuracy's to understand how quickly the networks are learning and to see whether they are learning information that generalizes well.

On the low frequency data, CCP-high learns much faster and reaches a higher train accuracy especially when the radius is larger. Looking at the validation accuracy shows that CCP-high is massively overfitting and performing below the validation accuracy of CCP-low. It is interesting that CCP-low in validation actually does better with a smaller radius (less information). It seems that overfitting is occurring with CCP-low as well and the more information it has, the worse it generalizes.

On the high frequency data, CCP-high learns much faster, especially as the radius shrinks (more information is given) and while it overfits, it does actually generalize a bit better than CCP-low when the radius is 8 and 12. In this case we corroborate the findings from Choraria et al. as a higher degree polynomial is able to learn more quickly and generalize better when given high frequency information.

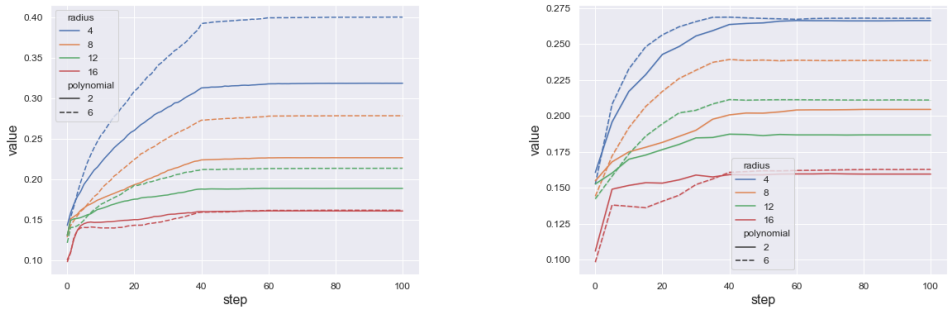
A question that arises is if this is truly the degree of the polynomial or related to the number of parameters as CCP-low has much fewer parameters than CCP-high. We conduct a further experiment where we take a CCP-low with the all the same hyperparams except changing the hidden size to 48. This means it has 295402 parameters which is comparable to CCP-high. However the learning curve looks identical to that of CCP-low with a hidden size of 16, which highlights the fact that it really is the polynomial degree which improves both learning and generalizability using high frequency information.



(a) Training Accuracy

(b) Validation Accuracy

Figure 6: CCP-high & CCP-low trained on low frequency filtered CIFAR10 data.



(a) Training Accuracy

(b) Validation Accuracy

Figure 7: CCP-high & CCP-low trained on high frequency filtered CIFAR10 data.

4 Re-Evaluating Hypothesis

We had shown that different polynomial degrees rely on spurious correlations by varying amounts and that a higher degree does slightly increase learning and generalization on high frequency components. However, we were still struggling to find a compelling case that demonstrated the polynomial degree altering the expressivity of the network. We went back to re-examine our hypothesis in light of current research.

Dynamically adjusting the polynomial expansion per data sample will increase the representation power of polynomial networks.

4.1 Current Research

There are three papers that discuss underlying properties that neural networks trained with gradient descent and cross-entropy loss all seem to share which cast some doubt on our initial hypothesis.

The first paper Shah et al. [2020] discusses simplicity bias, which is the idea that training procedures such as SGD seem to find simple models. (Note: There also seems a connection between simplicity bias and spectral bias, which is the tendency of neural networks to learn low-frequency functions first.) At first glance, this seems like a positive property and are postulated for being the reason why over-parameterized networks that can memorize datasets and obtain 100% training accuracy still generalize well. However, the paper shows that this bias can be extreme, to the point that complex models with different architectures will use only the simplest predictive feature, even when a complex feature is more predictive. They postulate that the simplicity bias is a unifying factor in key failure modes of deep learning such as poor out of distribution performance, adversarial vulnerability, and sub-optimal generalization.

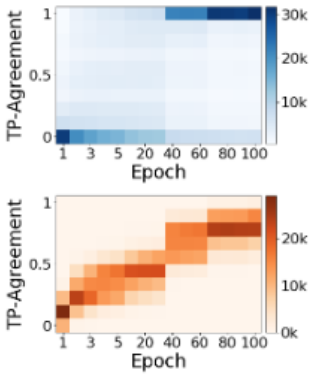
The second paper Nakkiran et al. [2019] studies the dynamics of SGD in learning deep neural networks and show that all the performance of the initial epochs can be explained by a linear classifier, and later epochs learn increasingly complex functions. The most important piece is the initial linear function that is learned is retained throughout training, even if training is taken to 100% test accuracy.

The last paperHacohen et al. [2020] demonstrates that deep neural networks learn examples in the training and test set in a similar order. They note the phenomenon is strongest for models of the same architecture, but it also crosses architectural boundaries.

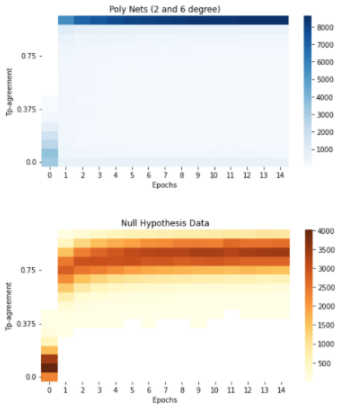
These three papers along with our experiments to date have prompted me to question our underlying hypothesis.

We performed a test to validate that the results of the third paper do hold with polynomial networks. When changing the degree of the polynomial the networks still learn the training set in a very similar order as shown in 8b.

If we were selecting between two networks that had learned the exact same function of the data, then there would be no point in making a decision between the two of them. Therefore, our hypothesis



(a) 27 models of Resnet-50 trained over ImageNet.Hacohen et al. [2020]



(b) 8 CCP 2 degree polynomials and 8 CCP 6 degree polynomials. TP agreement on validation set.

Figure 8: Top row in blue is TP-agreement on the validation set. Bottom row in orange is the null hypothesis TP-agreement based on random assignment of lables to images that give the same accuracy at the same

relies on the fact that adjusting the degree of the polynomial changes the expressivity of the network in some way. The above 3 papers provide evidence that neural networks trained using SGD on the same data will learn the same simple function (that which a linear classifier would learn), will keep this same simple function as a sort of anchor as it learns more complex features, and will learn the dataset in the same order. These attributes were seen more strongly for models with similar architectures.

We can see that we are grappling with problems that people in the ensemble literature have been looking at for years. In changing the degree we need to make two assumptions:

Assumption 1 *Models of different degrees are diverse (they make different errors and are better in certain areas of the dataset)*

Assumption 2 *The models of different degrees are all accurate (they are not dominated by one another)*

In testing our hypothesis we used SGD and cross-entropy loss and only varied the degree of the polynomial. Our experiments agreed with the above papers as varying the degree of the polynomial did not change the expressivity of the network. For assumption 1, it is clear that solely changing the degree of the polynomial is not enough to promote diverse models.

Thinking about our problem in terms of the ensemble literature is helpful and we can draw on the research that has been done in that field. Gontijo-Lopes et al. showed that model pairs that diverge more in training methodology display different generalization behavior and produce increasingly uncorrelated errors. The divergence is larger as we move further down this list:

1. initialization
2. hyperparameters
3. architectures
4. frameworks
5. datasets

They note that low accuracy models can be useful if they are trained differently enough. Diversity is not a necessary condition to have an improvement in accuracy, but it is important if you want to see meaningful improvement that is not just reducing the noise of the models.

We wanted to understand our polynomial networks and how they would look in their framework. For example changing the polynomial degree can be thought of as changing the architecture (slightly) and would seem to offer some benefit in an ensemble framework.

4.2 Ensemble Experiment

There are two main goals in creating ensembles of polynomial networks. The first is to compare the logits of the models on the test set to understand how diverse these models are. This would be a final test to see if the degree of the polynomial does enforce an implicit bias that changes the diversity of the function learned. The second is to have a baseline against which we can compete against and try to improve upon with a dynamic network.

For this experiment we trained two sets of ensembles. The first ensemble was created by training 4 separate models, each with different degrees of polynomial. The second ensemble was trained according to Huang et al. [2017] where we trained one model with a cyclic annealing learning rate and took snapshots of the weights to create 4 different models. The model used in all cases is based on the NCP-decomposition that has been modified to look like the ResNet architecture. It does not use activation functions, but does use batch normalization and instance normalization to stablize training. The training hyperparameters can be seen in 4 and the dataset is CIFAR100.

Table 4: Ensemble 1 and 2 Training Parameters

	degree	Res-Net Blocks	parameters	hyperparams	opt
NCP-low	2^4	[1, 1, 1, 1]	670216	epochs: 50 batch size: 128 n_channels: [64, 128, 256, 512]	SGD momentum: 0.9 Step LR lr: 0.1 milestone: [15, 25, 35, 45] gamma: 0.1
NCP-mid1	2^6	[1, 1, 2, 2]	5687466		
NCP-mid2	2^6	[1, 1, 2, 2]	11526570		
NCP-high	2^8	[2, 2, 2, 2]	11916460		

	degree	Res-Net Blocks	parameters	hyperparams	opt
NCP-high	2^8	[2, 2, 2, 2]	11916460	epochs: 200 batch size: 128 n_channels: [64, 128, 256, 512]	SGD momentum: 0.9 Cosine Annealing Warm Restarts LR lr high: 0.2 lr low: 0.0 restart: 50

The results of the individual models are shown in table 5. The ensemble was created by first heating up the temperature of the softmax using temperature scaling Guo et al. [2017] and taking an average. In ensemble 1 the improvement from the best individual model is 7.85% and in ensemble 2 the improvement is 7.14%. The ensemble is a clear improvement over the individual models.

What we are interested in is how diverse the individual models are and if there is more to the ensemble improvement than reducing variance among the predictions. We use the framework developed in Gontijo-Lopes et al. to answer this question. When comparing two models we can look at a metric θ which is an angle between -45 and $+45$ degrees and represents the confidence of the two models in

Table 5: Test Accuracy on CIFAR100

NCP-low	NCP-mid1	NCP-mid2	NCP-high	**ensemble 1**
.4214	.4292	.4336	.4308	**.5121**

NCP-high0	NCP-high1	NCP-high2	NCP-high3	**ensemble 2**
.4597	.4758	.4277	.4114	**.5472**

comparison to the other (fig 9). If the model 1 is very confident and model 2 not confident, then the degree will be close to $+45$ degrees and likely model 1's vote will win in the ensemble average.

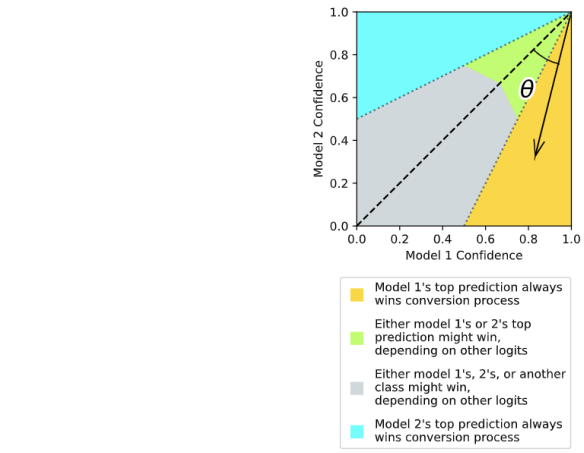


Figure 9: Explanation of the metric theta used in the following plots. Taken from Gontijo-Lopes et al.

Using this metric we can look at the distribution of θ over the test set for two models. In this case the distribution of θ is a proxy for diversity. For two models to be diverse we want them confident on different samples in the dataset. In the four plots this would manifest itself differently depending on how we subset the samples. For samples where Both Models Correct, a distribution that looks more uniform would symbolize a larger diversity in confidence. For samples where Neither Model Correct, more diverse models would cover a larger set of the dataset and therefore we would want Gaussian distribution to shrink as diversity increased. In Only Model 2 Correct we would want the distribution to be heavily skewed towards the model 2 confident side (and vice versa for Only Model 1 Correct).

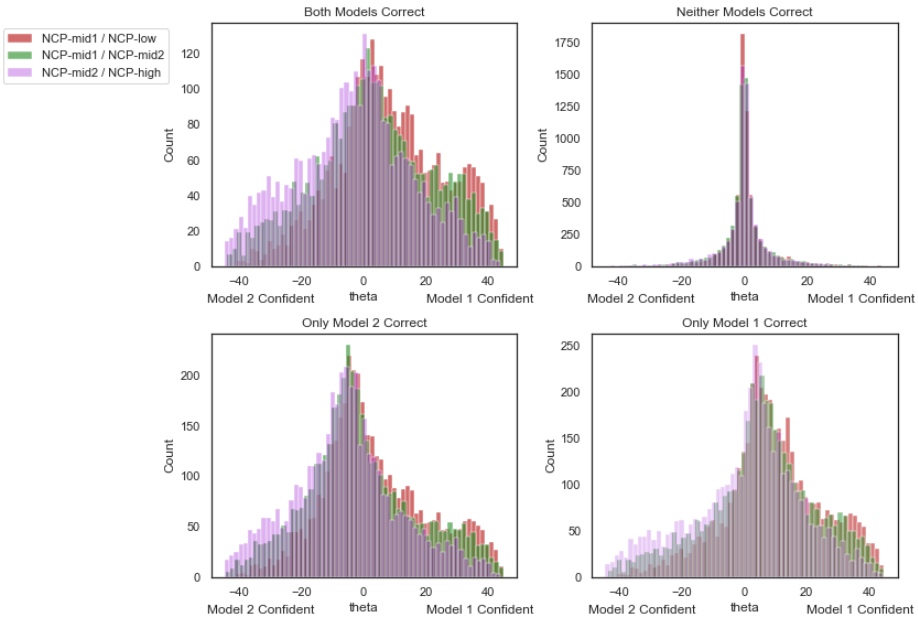


Figure 10: Each plot represents the distribution of θ across a different subset of samples. This subset of samples is denoted by the title of the plot. The color represents a distribution built by computing θ across different models with the models denoted in the legend Model 1 / Model 2.

These plots corroborate our earlier findings that changing the degree of the polynomial does not result in more diversity of the result. In fig 10 we compare the models of varying degrees and do not

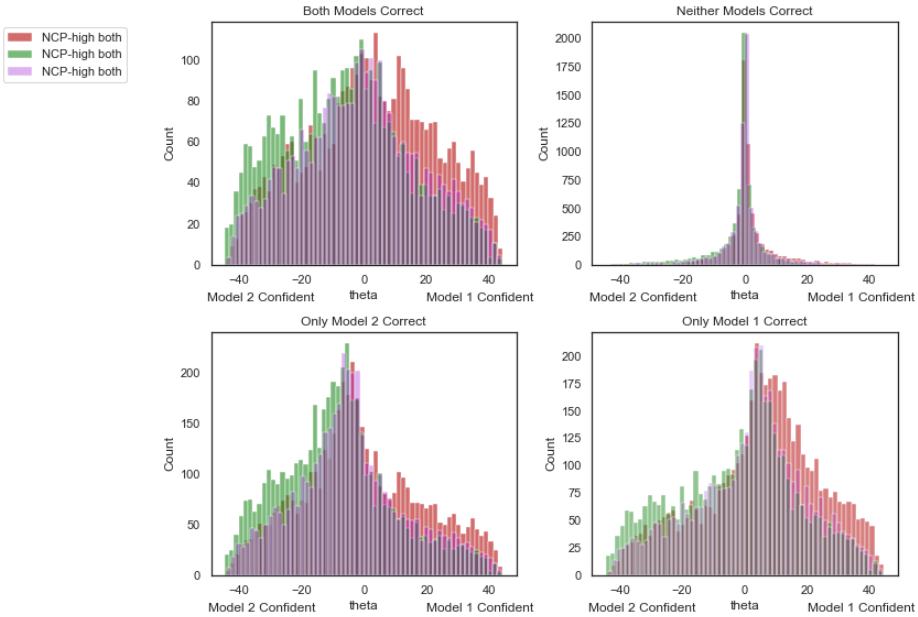


Figure 11: Each plot represents the distribution of θ across a different subset of samples. This subset of samples is denoted by the title of the plot. The color represents a distribution built by computing θ across different models

see the diverse distributions as we laid out above. What we see are distributions that look similar to Gontijo-Lopes et al. when the two models compared are the same models with reinitialized weights.

5 Narrowing Goals

Because our hypothesis does not hold, we will have to scale back our original ambitions. What would still be useful is to create a model that obtains the same effect of reducing variance as an ensemble, without paying the cost of training an ensemble. One technique that accomplishes this from is from Huang et al. [2017], however at inference time the data needs to be passed through M models and there is still engineering complexity in taking these snapshots and raising the temperature. A worthwhile goal then would be to design a polynomial network that achieves the same results as an ensemble but all in one model. To do this we will leverage the polynomial degree. While we have shown that changing the degree doesn't promote diversity, we can still use the concept of changing the degree to design a network that reduces variance of the output in a similar manner to an ensemble. This will be attempted as future work given the time constraints of the project.

6 Conclusion

We started out with the goals of making better neural networks by using the polynomial framework. Our ambitious goal was to try and make a dynamic network that changed the polynomial degree based on every sample. To do so we formulated a hypothesis which we wanted to prove out before attempting to design such a network.

Dynamically adjusting the polynomial expansion per data sample will increase the representation power of polynomial networks.

However, after a few initial experiments we were still unable to experimentally find an example when changing the degree changed significantly the function being learned. Diving back into the literature, we found discouraging signs that just changing the degree of the polynomial wouldn't be enough to encourage diversity in the function learned. We conducted further experiments and experimentally confirmed that different degree polynomials were learning very similar functions. While this invalidates our hypothesis, it means that we would have to change more than just the polynomial degree to get the diversity that we wanted.

References

Devansh Arpit, Stanisław Jastrz̨, Nicolas Ballas, David Krueger, Emmanuel Bengio, Maxinder S Kanwal, Tegan Maharaj, Asja Fischer, Aaron Courville, Yoshua Bengio, and Simon Lacoste-Julien. A closer look at memorization in deep networks. 2017.

Rishi Bommasani, Drew A. Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S. Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, Erik Brynjolfsson, Shyamal Buch, Dallas Card, Rodrigo Castellon, Niladri Chatterji, Annie Chen, Kathleen Creel, Jared Quincy Davis, Dora Demszky, Chris Donahue, Moussa Doumbouya, Esin Durmus, Stefano Ermon, John Etchemendy, Kawin Ethayarajh, Li Fei-Fei, Chelsea Finn, Trevor Gale, Lauren Gillespie, Karan Goel, Noah Goodman, Shelby Grossman, Neel Guha, Tatsunori Hashimoto, Peter Henderson, John Hewitt, Daniel E. Ho, Jenny Hong, Kyle Hsu, Jing Huang, Thomas Icard, Saahil Jain, Dan Jurafsky, Pratyusha Kalluri, Siddharth Karamcheti, Geoff Keeling, Fereshte Khani, Omar Khattab, Pang Wei Koh, Mark Krass, Ranjay Krishna, Rohith Kuditipudi, Ananya Kumar, Faisal Ladhak, Mina Lee, Tony Lee, Jure Leskovec, Isabelle Levent, Xiang Lisa Li, Xuechen Li, Tengyu Ma, Ali Malik, Christopher D. Manning, Suvir Mirchandani, Eric Mitchell, Zanele Munyikwa, Suraj Nair, Avanika Narayan, Deepak Narayanan, Ben Newman, Allen Nie, Juan Carlos Niebles, Hamed Nilforoshan, Julian Nyarko, Giray Ogut, Laurel Orr, Isabel Papadimitriou, Joon Sung Park, Chris Piech, Eva Portelance, Christopher Potts, Aditi Raghunathan, Rob Reich, Hongyu Ren, Frieda Rong, Yusuf Roohani, Camilo Ruiz, Jack Ryan, Christopher Ré, Dorsa Sadigh, Shiori Sagawa, Keshav Santhanam, Andy Shih, Krishnan Srinivasan, Alex Tamkin, Rohan Taori, Armin W. Thomas, Florian Tramèr, Rose E. Wang, William Wang, Bohan Wu, Jiajun Wu, Yuhuai Wu, Sang Michael Xie, Michihiro Yasunaga, Jiaxuan You, Matei Zaharia, Michael Zhang, Tianyi Zhang, Xikun Zhang, Yuhui Zhang, Lucia Zheng, Kaitlyn Zhou, and Percy Liang. On the opportunities and risks of foundation models. 8 2021. doi: 10.48550/arxiv.2108.07258. URL `https://arxiv.org/abs/2108.07258v2`.

Moulik Choraria, Leello Dadi, Grigorios Chrysos, Julien Mairal, and Volkan Cevher. The spectral bias of polynomial neural net-works. URL https://openreview.net/pdf?id=P7FLfMLTSEX.

Grigorios G Chrysos, Stylianos Moschoglou, Giorgos Bouritsas, Yannis Panagakis, Jiankang Deng, and Stefanos Zafeiriou. ´nets: Deep polynomial neural networks.

Grigorios G Chrysos, Markos Georgopoulos, Jiankang Deng, and Yannis Panagakis. Polynomial networks in deep classifiers. 2021.

Raphael Gontijo-Lopes, Yann Dauphin, and Ekin D Cubuk. No one representation to rule them all: Overlapping features of training methods.

Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q Weinberger. On calibration of modern neural networks. 2017.

Guy Hacohen, Leshem Choshen, and Daphna Weinshall. Let's agree to agree: Neural networks share classification order on real datasets. 2020.

Yizeng Han, Gao Huang, Shiji Song, Senior Member, Le Yang, Honghui Wang, and Yulin Wang. Dynamic neural networks: A survey.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2016-December:770–778, 12 2015. ISSN 10636919. doi: 10.48550/arxiv.1512.03385. URL https://arxiv.org/abs/1512.03385v1.

Jie Hu, Li Shen, Samuel Albanie, Gang Sun, and Enhua Wu. Squeeze-and-excitation networks. URL http://image-net.org/challenges/LSVRC/2017/results.

Gao Huang, Yixuan Li, Geoff Pleiss, Zhuang Liu, John E. Hopcroft, and Kilian Q. Weinberger. Snapshot ensembles: Train 1, get m for free. *5th International Conference on Learning Representations, ICLR 2017 - Conference Track Proceedings*, 4 2017. doi: 10.48550/arxiv.1704.00109. URL https://arxiv.org/abs/1704.00109v1.

Preetum Nakkiran, Gal Kaplun, Dimitris Kalimeris, Tristan Yang, Benjamin L. Edelman, Fred Zhang, and Boaz Barak. Sgd on neural networks learns functions of increasing complexity. *Advances in Neural Information Processing Systems*, 32, 5 2019. ISSN 10495258. doi: 10.48550/arxiv.1905.11604. URL https://arxiv.org/abs/1905.11604v1.

Harshay Shah, Kaustav Tamuly, Aditi Raghunathan, Prateek Jain, and Praneeth Netrapalli. The pitfalls of simplicity bias in neural networks. 6 2020. URL http://arxiv.org/abs/2006.07710.

Siyuan Shan, Yang Li, and Junier B Oliva. Meta-neighborhoods. URL https://github.com/lupalab/Meta-Neighborhoods.

Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in Neural Information Processing Systems*, 2017-December:5999–6009, 6 2017. URL https://arxiv.org/abs/1706.03762v5.

Haohan Wang, Xindi Wu, Zeyi Huang, and Eric P Xing. High-frequency component helps explain the generalization of convolutional neural networks. a.

Xiaolong Wang, Ross Girshick, Abhinav Gupta, and Kaiming He. Non-local neural networks. b.

Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. *Communications of the ACM*, 64:107–115, 11 2016. ISSN 15577317. doi: 10.48550/arxiv.1611.03530. URL https://arxiv.org/abs/1611.03530v2.