Research on Infinite Neural Networks

Shangyang Min and Zhilong Feng
Department of Computer Science
Michigan State University
East Lansing, MI

Abstract

As the research in deep learning goes further, the way to overcome the limitations of traditional deep learning using a finite neural network with fixed width is being more noticed. We are developing infinite neural networks to overcome problems that typical deep learning models cannot be easily solved. By using infinite neural networks, we find the potential opportunity for the theory of infinite neural networks to improve the solutions for image processing in real-life problems and how they can contribute to the deep learning community.

1 Introduction

As we know, we are at an age of artificial intelligence, and the newly developed theory - *Deep Learning* is running in the lead to open up the black box of AI (Castelvecchi, 2016). With the limitations of deep learning, the recent developed library *Neural Tangents* (Novak et al., 2019) gives us the ability to build infinite networks based on the theory of *Bayesian Infinite Neural Network* (Novak et al., 2018) and *Gradient Descent Trained Infinite Network* (Jacot et al., 2018).

Few previous works are building on the theory of infinite neural networks, and what we are doing is rare and new. There is research comparing the finite neural network and infinite neural network by theory and architecture, but infinite neural networks are still debated when applied to real-life tasks. Our purpose of this project is to find the potential of infinite neural networks through experiments and see what infinite neural networks can bring to the field of deep learning by comparing the results of baseline finite neural networks with it.

In our experiment, we want to build a baseline model based on a conventional neural network structure that will filter the images and is most ideally suited to image processing. The idea of our project is to build models with a similar training power on the same dataset.

The infinite neural network is an idea worthy of further exploration. One of the neural network's limitations is the width limitation. (Nguyen et al., 2020) The common knowledge in neural networks is that the performance of the model will change by varying the width and depth. Increasing either width or depth will typically increase the power of the neural network, but there is a problem coming from higher number of depth. With a higher number of depth, the network is harder to train, and it will most likely overfit the training data. So we are curious about what will happen when the depth remains the same, and we increase the width as we train. The infinite neural network gives us a direction to explore the answer and overcome the problem faced in deep learning.

Since there are few ongoing research projects on the theory of infinite-width neural networks, most of the publications debate the advantages and disadvantages behind its infinite-width theory. In this paper, we are testing and evaluating the potential probability of the theory by signing them to a real-life task. We are hoping to find a better modeling pipeline through it by comparing it with the baseline model. In the paper, we want to examine the infinite neural network's performance in the image processing category. By doing this, if the results are appreciated, we can significantly improve the training strategy of all the image processing tasks and help contribute to the community.

2 Our Team

We are a small person's team, and our responsibility is to communicate with each other about the challenges we will face: writing the training code, processing data, exploring the newly developed library *Neural Tangents*, constructing baseline model, constructing the infinite neural network, evaluate the models, and compare the performances between the conventional neural network with limited width and the conventional neural network implemented

using an infinite width.

Shangyang Min net Id *minshang*, Department of Computer Science, Department of Communication Arts. Senior in the program.

Zhilong Feng MSU netId *fengzhi2*, Department of Computer Science. Senior in the program.

3 Related Works

There are already a group of talented engineers at Google who have developed a model called *Neural tangents* (Novak et al., 2019). This is a high-level neural network API for specifying complex, hierarchical neural networks of both finite and infinite width. Neural Tangents allow researchers to define, train, and evaluate infinite networks as quickly as finite ones.

Neural Tangents allows us to construct a neural network like other libraries with the functionality like convolutions, pooling, residual connections, nonlinearities, etc. , and obtain not only the finite model, and more importantly the kernel function of the respective Gaussian Processes which are the theory of infinite neural network based on(Neal, 1996).

The engineers in Google developed this library in python using JAX(Bradbury et al., 2018) and leveraging XLA (a domain-specific compiler in TensorFlow), making it possible to run out-of-the-box on CPU, GPU, or TPU. It also has near-perfect scaling ability with highly optimized code for speed and memory efficiency. This is the library we will mainly use in this research.

4 Dataset

For this research, we will use Animals-10 (Alessio, 2019) dataset from kaggle.com [1]. It is a dataset uploaded by Corrado Alessio [2] 3 years ago, and it already has more than 167k viewers and 28k downloads.

It contains about 28K medium-quality animal images with GPL 2 license. These images belong to 10 categories: dog, cat, horse, spyder, butterfly, chicken, sheep, cow, squirrel, and elephant. All these images have been collected from "Google images" and checked by humans. Meanwhile, there is some manually added error data to simulate actual conditions.

[1]Kaggle, a subsidiary of Google LLC, is an online community of data scientists and machine learning practitioners.

[2]Corrado Alessio, alessiocorrado99@gmail.com

All those images in the dataset are divided into folders, one for each category. Each category will contain nearly 2k to 5k images. We plan to use 70% of each species for training, 15% for validation, and the rest 15% for testing.

The author has tested this dataset and got nearly 80% accuracy with CNNs.

5 Methodology

Since there are no significant features we can extract from the images, we are not planning to use a basic fully-connected DFNN model, which we will cover in class. Instead, we will use a conventional neural network as a baseline approach model for comparison, which we will not cover in the lecture. Based on that, we will use Neural Tangents to train the infinite neural networks with the same dataset and to see if there is any improvement in the accuracy and loss.

In the infinite neural network training, we will train two models with the best performance based on the two theories of Bayesian infinite neural network (or the NNGP kernel) (Xu et al., 2021) (which the purpose is to fit the network to data by maximizing the lower bound on marginal likelihood given by the infinite-dimensional ELBO equation 1) and gradient descent trained infinite neural network (Jacot et al., 2018) (which during training the network function evolve along the negative kernel gradient with respect to the neural tangent kernel (NTK) 2). We will optimize the performance by using and finding different hyper-parameters, topologies, and model structures within a similar power.

$$\mathcal{L}_{\mathrm{ELBO}_\infty}(\phi) = \mathbb{E}_{q_\phi(w)}\left[\log p(\mathcal{D}|w) - \int_0^1 \tfrac{1}{2}\,\|u(w_t, t, \phi)\|_2^2\,\mathrm{d}t\right].$$

Figure 1: ELBO equation, which is the function we are maximizing in NNGP infinite neural network approach

$$\Theta^{(L)}(\theta) = \sum_{p=1}^{P} \partial_{\theta_p} F^{(L)}(\theta) \otimes \partial_{\theta_p} F^{(L)}(\theta).$$

Figure 2: Core equation using in NTK Infinite neural network approach

As described in the previous section, we randomly select 70% of images in each folder, which means different species, and use these data to train under the CNN and Neural Tangents models. After that, we randomly select half of the left 30% for

validation. Finally, we will use the last 15% to test these two models and compare the results.

6 Experiment I

6.1 Preparation

On the data preparation phrase, we summarized and divided the animal data into 10 different outcome labels (see Table 1). The images are loading using *CV2* package, and we resized to 224×224 (see the section 6.5 below and we will test the influences of the performances by varying the resolution size).

Species	Count	Species	Count
Sheep	1820	Dog	4863
Squirrel	1862	Butterfly	2112
Cat	1668	Spider	4821
Horse	2623	Elephant	1446
Cow	1866	Chicken	3098

Table 1: The whole animal-10 dataset into 10 species, the dataset is most likely evenly divided, with a higher sample number of dog and spider images

6.2 Models

We built three conventional neural networks: (1) A traditional neural network based on the TensorFlow Keras library and (2) an NTK Infinite neural network based on the Neural Tangent library, (3) an NNGP Infinite neural network based on the Neural Tangent library. For a fair comparison, we constructed the network using the same filter number of the conventional layer with its corresponding index of depth number, and they are built in the same depth. The activation function used in the network is simple $Relu$.

6.3 Training

In the training section, the traditional CNN is trained with random initialization and Adam optimizer. We are using the early-stopping method provided by the TensorFlow library with a patient number equal to 5, and we save the model with the lowest MSE training loss in the validation group. The infinite-width neural network does not provide a training interface, and it does not provide a way to see the performance changes in the validation set during the training. In this situation, after we define the kernel training strategy, we input our testing data for the prediction Y label.

6.4 Result

Table 2 is the result of our first experiment. We use Python *time.perf_counter* to count the time of training cost for each model since it has higher precision than the normal counter, and it is usually used for run time count. As we see from the result, the traditional CNN has a lower time cost than the other two infinite-width network approaches. An important thing is we noticed in the Neural Tangent GitHub issue that the library developer notifies that there is not a layer with *softmax* (more discussed in section 6.5) function we can add to the end of the network to do the classification in infinite-width network approaches because it is doing poorly. The accuracy score we attended is through evaluating the testing set by getting the index with the max possibility of the input image. After getting the index of max value in each image for both test labels and predicted labels, we recorded if the two labels were matched, if they were correctly matched, we weight it as 1, else 0. Then we calculate the mean of the added weights based on the whole testing dataset. By comparing the mean of accuracy scores based on repeating the experiments, the two types of infinite width neural network is outperformed by traditional neural network.

Models	Average Time cost(s)	Accuracy score
CNN	267	0.513
NTK	313	0.361
NNGP	281	0.272

Table 2: The table records average of time cost and accuracy score for baseline CNN, NTK, and NNGP

6.5 Improvements and Problems

6.5.1 Size of images

In the experiment, we resized the images to 224*224 to save memory. This means we will drop many pixels and features from the original figure. We would like to know if this is one of the reasons that can have more influence on the infinite width approach than the traditional CNN approach in accuracy values. In the further steps, we will try to use a lower resolution like 64*64 or 96*96 or a higher resolution like the full-size picture (Figure 3) to see if this will have different impacts on the model performance. Of course using larger images will cause the *NotEnoughMemory* problem. We are planning to solve this problem by changing the batch size (see 6.5.2).

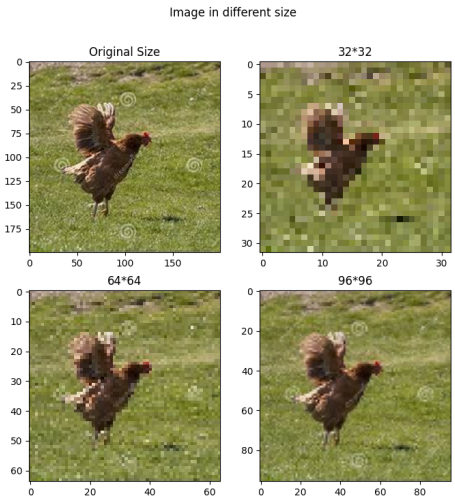


Figure 3: Images in original size, 32×32, 64×64, and 96×96 resolution

6.5.2 Size of batches

For this experiment, we set the batch size to 76 (1/24 of the test dataset), which is the largest size we could reach without a *NotEnoughMemory* warning. A core idea we get from the theory of the infinite-width neural network is that we need to fit all the data by extending the width, which takes around 34TB in the structure we built, which is impossible to do. It may lower the performance since we are using image data that takes a large space of memory.

6.5.3 Maxpooling

Is maxpooling the problem that causes the infinite-width neural network to fail in the comparison accuracy? The neural tangent does not have the functionality to add maxpooling layer when constructing the CNN model. Since this library relies on JAX, we may find a solution in the next result-approaching strategy through the original library.

6.5.4 Softmax Function

The neural tangent library does not provide a softmax function because it theoretically does not fit in the infinite-width neural network. However, it does provide a sigmoid-like function we can use for binary classification. In the following experiment, we may choose two labels of outcome animals with similar sample sizes and train the three models from 6 to perform a binary classification task. This way, we may get a more fair evaluation score.

7 Discussion of Experiment I

From the experience I results, we are considering rebuilding the structures of our networks and applying different training strategies based on the results. We are assuming the result caused is by the lack of features provided by the neural tangent library and its original Jax library. We are going to explore further the original library and make the adjustment to the structure of the network, and use different training hyper-parameters to perform our next experiment coming up.

8 Experiment II

In Experiment I, we have our primary results that in imaging processing, the Infinite Width Neural Network may not perform well like the traditional CNN. And we provide issues that associate with the structural issues of the infinite-width neural network and the library issue provided by Neural Tangents.

In Experiment II, we address the questions we had in Experiment I and find the potential improvements with these issues. We also changed our strategy in order to make a more detailed comparison with limited resources(described in 8.1). Instead of training the best models to apply in the animal classification tasks, we focus more on the comparison aspect of the two models, so we decrease the power of the two models (while keeping them in similar power in comparison). Below I will describe each issue we faced during Experiment II and our solution.

8.1 Library Lack of Compatibility

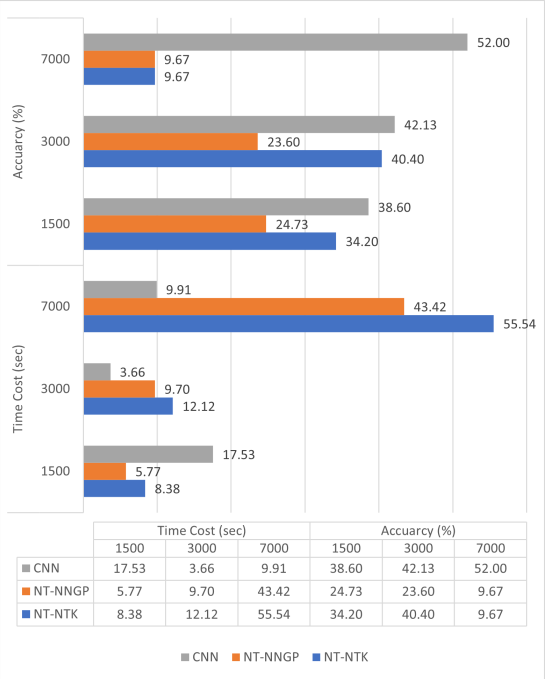
In Experiment I, We were using the MSU HPCC platform and using a V100 graphic card to do the training experience. Recently HPCC has an upgrade to its CPUs and GPUs, the new A100 graph cards are replaced, the CUDA version of 11.5 is locked in the platform, and the Neural Tangent is not compatible with that CUDA version.

The solution we made is to change our training platform to Google Colab, it has faster training speed and lower memory resource so the results in Experience II is not comparable with Experiment I.

8.2 Bad performance with large sample size

As in figure 4, we found out when we feed the whole training dataset that in our training strategy,

both infinite width neural networks are having terrible performance when we apply them to the testing data. So we shuffled to make the data still evenly divided and chose 3000 (nearly half of the whole dataset) training data instead of a whole training dataset to eliminate this problem.



	Time Cost (sec)			Accuarcy (%)		
	1500	3000	7000	1500	3000	7000
CNN	17.53	3.66	9.91	38.60	42.13	52.00
NT-NNGP	5.77	9.70	43.42	24.73	23.60	9.67
NT-NTK	8.38	12.12	55.54	34.20	40.40	9.67

Figure 4: Difference of Time Cost and Accuracy based on different sample sizes used by CNN, NNGP, and NTK with an image size of 96*96 and batch size of 50

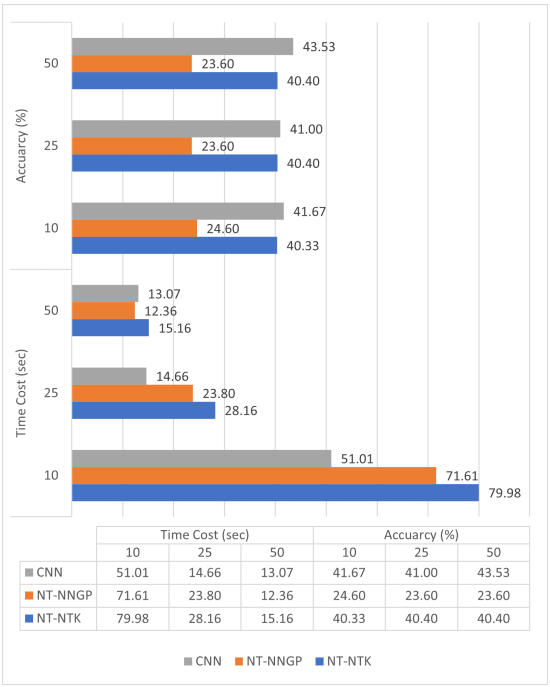
8.3 Output Layer Activation Function

As we discussed theoretically, the Softmax function is not working in infinite-width NN. In the library, they provided a function called Sigmoid_like function $f(x) = .5 * erf(x/2.4020563531719796) + .5$, which in our case is also not workable after we added it in our code.

After our consideration, we keep the strategy because the Softmax function is a classified function with the equation $probability = exp(value)/sumvinlistexp(v)$. We are still evaluating the performance using the max probability that returned in that axis.

8.4 Batch Issue

Since we doubted that batch size would influence the performance issue, there was no way we could read all images into memory in infinite-width neural network training. And the batch size is required to be evenly divided by the data size, we tested the training accuracy and time for training using different batch size numbers that can fit into the model. We recorded and calculated the average performance during the 10 times of training. Below is the table showing that (Figure 5):



	Time Cost (sec)			Accuarcy (%)		
	10	25	50	10	25	50
CNN	51.01	14.66	13.07	41.67	41.00	43.53
NT-NNGP	71.61	23.80	12.36	24.60	23.60	23.60
NT-NTK	79.98	28.16	15.16	40.33	40.40	40.40

Figure 5: Difference of Time Cost and Accuracy based on different batch sizes used by CNN, NNGP, and NTK with an image size of 96*96

8.5 Image Resolution

As we experienced in Experiment I, Infinite Width Neural Network's training strategy requires much more memory space than a traditional neural network. So based on the resources we have, we resized these images into $32 * 32$, $64 * 64$,$96 * 96$, $128 * 128$, and $160 * 160$, and we tested with appropriate batch size to do the training with the limited memory resource.

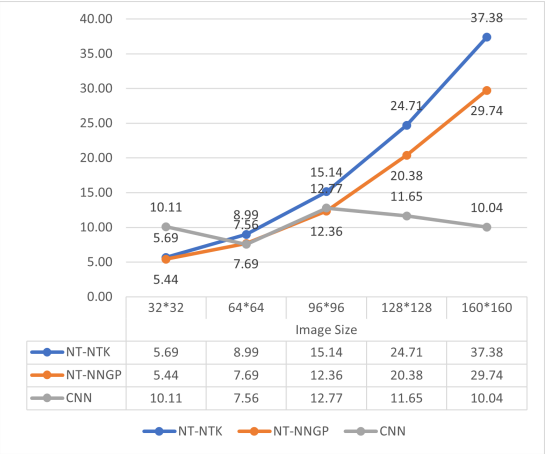


Figure 6: Difference of Time Cost based on different image resolution used by CNN, NNGP, and NTK with a batch size of 50

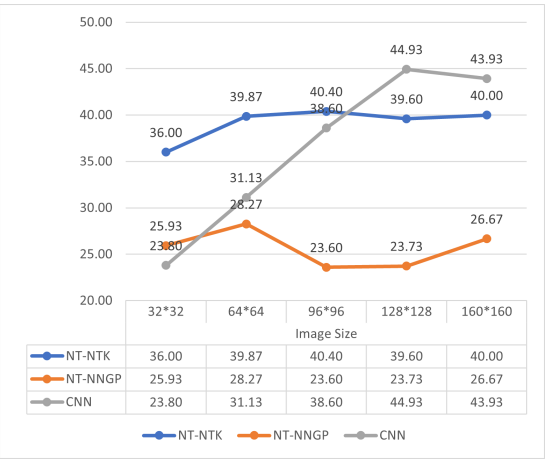


Figure 7: Difference of Accuracy based on different image resolution used by CNN, NNGP, and NTK with a batch size of 50

8.6 Down Sampling

As we mentioned, it is hard to downsample data after conventional layers in Neural Tangent. We found out that the average pooling is implemented in Neural Tangent. So we tested the infinite-width neural network with corresponding CNN using the average pooling.

By the figure 9, we found that traditional CNN increases its accuracy very fast in this situation. While neural tangent does not provide a way for us to check the progress. We tested twice on infinite-width CNN with average pooling, and they both crashed after 30 minutes of training because of memory allocating issues.

To solve this issue, we changed the training strategy of downsampling. Instead of using pooling, we tried to use strides=2 in comparison. And the graph shows the performance of the three different models (Figure 8).

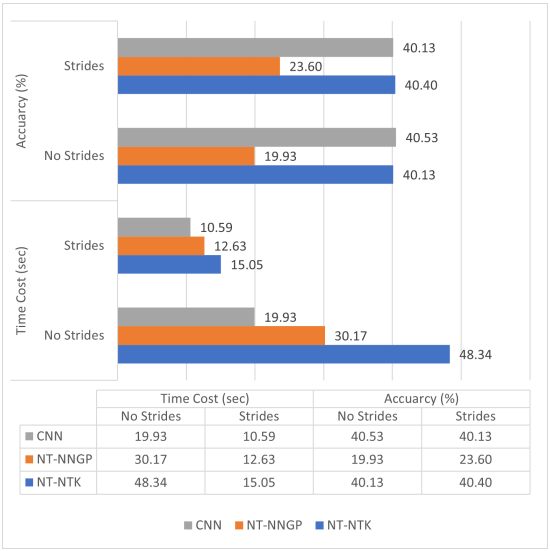


Figure 8: Difference of Time Cost and Accuracy based on applying or not applying strides on CNN, NNGP, and NTK using an image size of 96*96 and a batch size of 50

After we realized pooling is a big issue in improving both the performance in CNN and IFNN, we also tried to design both models without pooling. Pooling is a layer that contains rich information and can increase the accuracy of the overall models. In infinite-width neural network, the structure without pooling requires significantly fewer resources so we applied the structure to the whole dataset. Figure 9 contains the records from training.

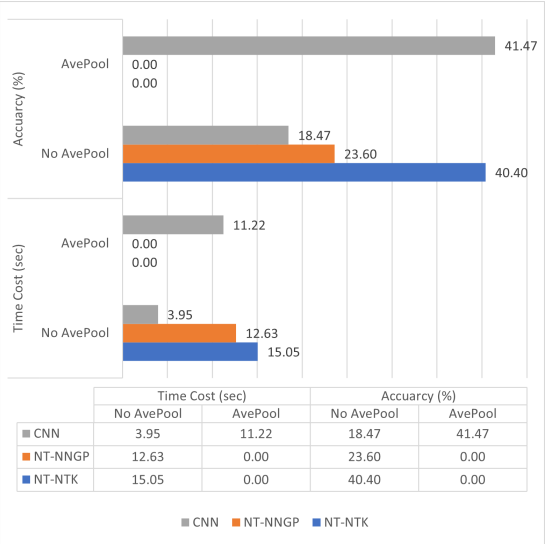


Figure 9: Difference of Time Cost and Accuracy based on applying or not applying AvgPooling on CNN, NNGP, and NTK using an image size of 96*96 and a batch size of 50. NT models do not have outputs when applying AvgPooling

8.7 Result

The infinite-width neural network had trouble when the training dataset was large. From figure 7, as the resolution of the image went higher, the accuracy had no significant influence in the infinite-width neural network, but in CNN the accuracy increased dramatically. The training time in the infinite-width neural network will linearly increase as there is less influence in traditional CNN. The batch size of the training in IFNN will increase the training time, and there is no significant influence on the actual accuracy result. From the down-sampling strategy, we tried methods to help the network do better feature extraction, and they demonstrate an obvious change from the result. By adding the average pooling, the infinite neural network failed in training, and pooling helps the accuracy in CNN a lot with nearly 200% increase in time. Downsampling with strides helps reduce the time in all three models but there shows no influence in accuracy.

9 Discussion for Experiment II

Overall, by summarising the comparisons between the infinite width conventional neural network and traditional conventional neural network in accuracy performance, training time cost, library support, and training resources. I am towards the downside of using an infinite neural network for real image tasking. By analyzing the results we get from the experiments, the infinite width neural network has overall lower in-class performance than the state-of-art conventional model with a similar training power. While the conditions remain the same. In 8, it reaches a similar accuracy with the CNN model but still requires nearly $2x$ time increasing, also the traditional CNN has better downsampling down sampling strategy rather than using the same strides method with infinite width neural network. In other experiences, the infinite width neural networks built by Neural Tangent all perform worse than the CNN model.

10 Future Research

10.1 Limitation

Our Research shows it is less practical to apply infinite neural networks in real-life tasks. By applying it in the image processing task, we do not see a significant improvement compared with traditional CNN, and it costs more resources to train than a traditional CNN model. A potential explanation for the lackluster performance of infinite networks reported in the literature is the absence of representation, or equivalently kernel learning, which results in less flexibility and consequently worse performance. (Aitchison, 2020) The lack of library support is also an issue the theory is facing. Neural Tangent compare with the commonly used library like TensorFlow by Google and Pytorch by Facebook. They support more functionally than Neural Tangent based on the Jax library. From our experience, it is harder to implement and use than the commonly used toolkit.

10.2 Direction

Infinite width neural network is a strong theory, we are excited about this theory to make changes in the general deep learning field of study. For its future, because we only tested it in image tasking and its functionality with conventional layers, and it is lack of potential to use the downsampling methods, commonly used activation functions like the way we commonly do in the traditional finite neural network. To conclude the rejection of infinite neural networks is too early. We are hoping to use this theory in other tasks like signal processing and NLP. To see if the infinite width neural networks can have the potential to outperform the state-of-act model in that field of research.

References

Laurence Aitchison. 2020. Why bigger is not always better: on finite and infinite neural networks. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 156–164. PMLR.

Corrado Alessio. 2019. Animal-10.

James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. 2018. JAX: composable transformations of Python+NumPy programs.

Davide Castelvecchi. 2016. Can we open the black box of ai? *Nature News*, 538(7623):20.

Arthur Jacot, Franck Gabriel, and Clément Hongler. 2018. Neural tangent kernel: Convergence and generalization in neural networks.

Radford M Neal. 1996. Priors for infinite networks. In *Bayesian Learning for Neural Networks*, pages 29–53. Springer.

Thao Nguyen, Maithra Raghu, and Simon Kornblith. 2020. Do wide and deep networks learn the same things? uncovering how neural network representations vary with width and depth.

Roman Novak, Lechao Xiao, Jiri Hron, Jaehoon Lee, Alexander A. Alemi, Jascha Sohl-Dickstein, and Samuel S. Schoenholz. 2019. Neural tangents: Fast and easy infinite neural networks in python.

Roman Novak, Lechao Xiao, Jaehoon Lee, Yasaman Bahri, Greg Yang, Jiri Hron, Daniel A. Abolafia, Jeffrey Pennington, and Jascha Sohl-Dickstein. 2018. Bayesian deep convolutional networks with many channels are gaussian processes.

Winnie Xu, Ricky T. Q. Chen, Xuechen Li, and David Duvenaud. 2021. Infinitely deep bayesian neural networks with stochastic differential equations.