Growing Neural Networks Achieve Flatter Minima

Paul Caillon, Christophe Cerisara

article

Growing Neural Networks Achieve Flatter Minima

Paul Caillon and Christophe Cerisara

Universite de Lorraine, CNRS, LORIA
54500 Vandoeuvre-les-Nancy, France
{paul.caillon, cerisara}@loria.fr

Abstract. Deep neural networks of sizes commonly encountered in practice are proven to converge towards a global minimum. The flatness of the surface of the loss function in a neighborhood of such minima is often linked with better generalization performances. In this paper, we present a new model of growing neural network in which we incrementally add neurons throughout the learning phase. We study the characteristics of the minima found by such a network compared to those obtained with standard feedforward neural networks. The results of this analysis show that a neural network grown with our procedure converges towards a flatter minimum than a standard neural network with the same number of parameters learned from scratch. Furthermore, our results confirm the link between flatter minima and better generalization performances as the grown models tend to outperform the standard ones. We validate this approach both with small neural networks and with large deep learning models that are state-of-the-art in Natural Language Processing tasks.

1 Introduction

Over the last few years, deep learning [20] has had empirical successes in multiple research domains, such as computer vision, speech recognition, and machine translation [11], [35], [28]. Along with its practical success, the theoretical properties of deep learning have been a subject of active investigation, from the expressivity [21], [1] and the generalization properties to the trainability [3], [15] of a network.

Some empirical works observe that generalization and flatness of the minima found during training are related [2], [33]. However, [7] questions this assumption, by showing that for deep neural networks with rectifier units, most Hessian-based measures of the flatness of the loss minimum are sensible to rescaling, making it possible to build equivalent models corresponding to arbitrarily sharper minima. To address this issue, a recent work [32] introduced a particular measure invariant to rescaling to show that flatter minima obtain better generalization performances than sharper ones. Sharper minima are thus believed to be suboptimal and should be avoided during learning. A recent work, [16] theoretically proves a related result, which is that adding one special neuron by output unit eliminates all suboptimal local minima of any neural network.

In Natural Language Processing, the models used in practice consist in millions of parameters. For example, BERT [6] is a well-known contextual word embeddings model that is pre-trained with a Denoising Autoencoding objective and is at the basis of most state-of-the-art results in many Natural Language Processing (NLP) tasks. We

study in this work, in the context of NLP and pattern recognition tasks, whether adding neurons incrementally instead of learning them all from the beginning could achieve a flatter minimum for the neural network. In the following, we propose to experimentally investigate this hypothesis by comparing an incrementally growing network with one with a fixed architecture learned from scratch, with both small and competitive models. The results we present tend to confirm our hypothesis as flatter minima are indeed achieved by growing networks, which also often leads to better generalization performances.

2 Related Work

Trainability of a Neural Network. Training neural networks can be seen as a non convex optimization problem. However because of the absence of poor local minima, the trainability of a Deep Neural Network is proven to be possible [15]. Recent results theoretically prove that gradient descent can find a global minimum for nonlinear deep neural networks of sizes commonly encountered in practice [17]. In fact a linear increase of the number of trainable parameters as the size of the dataset increases is sufficient to find a global minimum. Such a network generalizes well on unseen test samples.

Growing Neural Networks. The properties of growing networks, notably with regard to training convergence, are more and more investigated. The authors of [16] for example add a special neuron by output unit in order to eliminate all suboptimal local minima of any deep neural network. This idea of a network adapting its architecture to the dataset it is trained on is also found in automated Neural Architecture Search (NAS, see [8]). Typical works in this field are [4] and [5], where the authors use a grow-and-prune training paradigm to iteratively add and remove units in order to achieve more compact networks with state of the art performances.

The idea of progressive growing of models has also been explored in the context of Generative Adversarial Networks (GAN) [10] to increasingly add new details as the training progresses. In [14], the authors start with low-resolution images, and then progressively increase the resolution by adding layers to the networks. This allows the model to learn increasingly finer scale detail, instead of having to learn all scales simultaneously.

This kind of growing approach can be used in continual learning [30], where the goal is to train models that must be capable of progressively learning knowledge over long time spans. One of the main challenges in continual learning is catastrophic forgetting, i.e., the fact that new information interferes with previously learned knowledge [26]. Recent works in this field showed that growing a small network both wider and deeper allows to learn accurate and relatively small networks that can prevent catastrophic forgetting, achieving state-of-the-art performances with methods such as Learn-to-Grow [23], Compact-Pick-Grow [13] or recently Firefly Neural Architecture Descent [39]. However the question as to why those growing neural networks reach state of the art performance while being smaller than most of the other competitive models (comparable test accuracy as the full model with only 4% of the full model's size in [39]) is still to be investigated, one idea being that when a network grows, the parameter space becomes larger and what was previously a local minima can become a saddle point and hence can be escaped, which yields a monotonic decrease of the loss. [36]

Smooth and Sharp minima. On top of these considerations, the abilities to generalize to unseen data is often linked to the flatness of the minima found during training in empirical works [2], [33] sharp minima leading to poorer generalization. In a recent work, [38] notably shows that smoothing out and eliminating sharp minima by perturbing multiple copies of the DNN by noise injection and then averaging these copies lead to an improvement of the generalization properties. Another equivalent idea is explored in [34], which proposes to smooth activations instead of using noise or progressively adding units during training. A very interesting idea recently developed to enhance the performances of deep neural networks is to track the sharpness of the loss model and to jointly optimize the loss and its sharpness [9]. In particular, this procedure, called Sharpness-Aware Minimization (SAM), seeks parameters that lie in neighborhoods having uniformly low loss and present empirical results showing model generalization improvement across a variety of benchmark datasets.

In order to correctly evaluate the flatness of minima, measures invariant to the rescaling issue pointed out in [7] now exist [31, 32] and can be used to evaluate the difference of flatness in minima between models of the same size. The goal of our work is to investigate whether growing a feedforward neural network (FNN) from scratch throughout the learning phase yields better loss surface properties at minima than learning a standard FNN, both having ultimately the same number of parameters, using the measure developed in [32].

3 Model Description

3.1 Notations

The following notations are inspired by [18]. We consider the general case of neural networks of any depth for k-class classification that can be topologically considered as a directed acyclic graph (DAG) with non-linearity functions such as ReLU, tanh or sigmoid. This includes any structure of feedforward neural network with or without fully connected layers and with potentially skip-connections.

Let $N = \{1, \dots, n\}$ be the set of neurons in a network. For two neurons $(i, j) \in N^2$, we note :

- $w_{j,i}$ the weight of the connection from j to i. If there is no connection, then $w_{j,i} = 0$. Note that the sparse $n \times n$ matrix $W = [w_{j,i}]_{1 \leq j,i \leq n}$ is constrained to define an acyclic graph;
- σ_i the activation function at the output of neuron i;
- $\pi(i)$ the set of parents of neuron i: $\pi(i) = \{j | w_{j,i} \neq 0\}_{1 \leq j \leq n}$
- $d(i)$ the depth of neuron i: it is the longest path to reach i from any input.

Every DAG has at least one topological ordering, which can be used to create a layered structure with possible skip connections as shown for example in [12] and [29]. There is thus an equivalence between the representation of a neural network either with depths or with layers, as the layer l is composed of all the neurons at depth l. Given an input vector x, we define the pre-activation of a neuron i at depth l recursively as

$$z_i(x, W) = \sum_{j \in \pi(i)} \sigma_j \left(z_j(x, W) \right) w_{j,i}.$$

3.2 Model presentation

Recent works [24] and [17] propose to insert neurons in order to avoid bad minima. We investigate next the impact on the loss surface when inserting neurons into the model progressively throughout the learning process.

More precisely, our work's main focus is to explore whether growing approaches intrinsically leads to flatter minima. We do not seek an increase in the model performances per se. We rather try to understand why the growing approaches lead to better results than the standard ones in the recent works. To the extent of our knowledge, no similar work on the sharpness of the loss surface of growing networks exist. That is also why our model does not rely on complex heuristics to decide how and when the model's size should be increased. We rather choose a simple approach, based on random picks as it has been shown in previous works [22, 27] that advanced NAS techniques are often only marginally superior to simple random search.

In our model, the neurons are inserted incrementally at a regular pace (i.e., with a constant time interval) during the learning phase, starting with a minimum number of parameters (just the input and output layers) in order to increase the number of learnable parameters with the number of epochs. Intuitively, our growing procedure is a naive insertion process: first, an existing neuron is randomly chosen and splitted to create a new child neuron, which inherits most of its children. The new neuron also gets new parent connections with existing neurons randomly chosen in the previous layers. More formally, we have initially:

- The set of neurons is $N = \{1, \dots, n_I, n_{I+1}, \dots, n\}$: the first n_I neurons are the input neurons, while the last $n - n_I$ neurons are the output neurons;
- The weight matrix W is symmetrical and is initially strictly equivalent to a feed-forward neural network with no hidden layer, with non-null transitions that are initialized randomly, e.g., with Glorot or uniform initialization;
- The depth of the input neurons is 0, and the depth of the output neurons is 1.

Then, we randomly choose a neuron from the current network, called the *primary parent*, as well as $I - 1$ other neurons on lower depths, which are just called the *parents*. The primary parent is randomly sampled among all existing neurons, as previous works [22, 27] found that advanced NAS techniques are often only marginally superior to simple random search. With each neuron insertion, the total number of parameters of the model is increased by I, as each link created between the new neuron and its parents adds a parameter. This process is iterated until we reach the desired number of parameters. Intuitively, insertion proceeds by splitting the primary parent, i.e., inserting a new neuron after the primary parent that inherits its children, as detailed in Algorithm 1.

The connection between the primary parent and the neuron inserted is stored in π^0 which is the set containing all of the pairs *(primary parent, neuron inserted)*. We initialize π^0 as the set containing all the connections between input and output neurons, considering that all input neurons are "primary parents" of all of the output neurons : $\pi^0 = \{(1, n_{I+1}), \dots, (1, n), \dots, (n_I, n_{I+1}), \dots, (n_I, n)\}$. In this way, we are able to track the multiple insertions and to grow the hidden layers of the network both deeper and wider. The insertion process is illustrated in Figure 1.

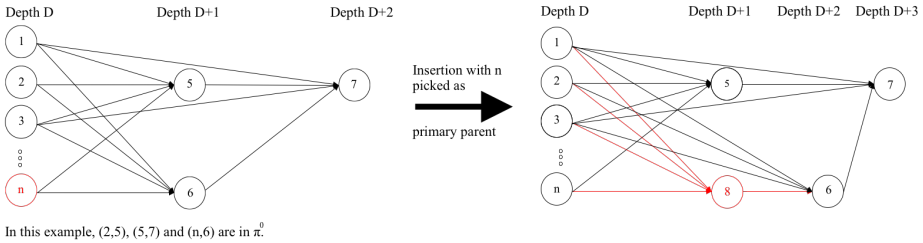

Fig. 1: Example of parents/children inheritance before and after insertion; the new weights are: $\forall i \in [1, n], w'_{i,8} \sim \mathcal{U}(-0.5, 0.5), w'_{8,6} = w_{n,6}$.

Algorithm 1: Insertion Algorithm

Input: Weight matrix $W \in \mathbb{R}^{\mathrm{n} \times \mathrm{n}}$

Output: Weight matrix $W' \in \mathbb{R}^{(\mathrm{n}+1) \times (\mathrm{n}+1)}$

Save the initial set of primary parents π^0

Create a new neuron $n + 1$

Create W' by copying all weights from W and initializing the new dimension with null weights.

Sample the primary parent: $j \sim \mathcal{U}(\{j | d(j) < D\}_{1 \leq j \leq n})$ where $\mathcal{U}$ is the uniform distribution, and $D = \max_{1 \leq i \leq n} d(i)$ is the depth of the network.

Save the new primary parent: $\pi^0 \leftarrow \pi^0 \cup \{(j, n + 1)\}$

Insert $n + 1$ by setting $w'_{j,n+1} \sim \mathcal{U}(-0.5, 0.5)$

Connect children of j as children of $n + 1$:

for $i \in \{1, \ldots, n\}$ *so that* $w_{j,i} \neq 0$ **do**

- if $d(i) > d(n + 1)$:
  - Set $w'_{n+1,i} = w_{j,i}$
  - Set $w'_{j,i} = 0$
- if $d(i) = d(n + 1)$ and $(j, i) \in \pi^0$:
  - Set $w'_{n+1,i} = w_{j,i}$
  - Set $w'_{j,i} = 0$
  - $\pi^0 \leftarrow \big(\pi^0 - \{(j, i)\}\big) \cup \{(n + 1, i)\}$

Add new parents for $n + 1$:

for *I-1 times* **do**

- Sample a node $k \sim \mathcal{U}(\{i | w'_{i,n+1} = 0)$ and $d(i) \leq d(j)\}_{1 \leq i \leq n})$.
- Add k as a parent of $n + 1$: $w'_{k,n+1} \sim \mathcal{U}(-0.5, 0.5)$
- Iterate

Remark 1. Steps 1 and 8 of the algorithm ensure that all the output units and only them are at maximum depth as when we insert a neuron n+1 on the maximal depth, n+1 is a primary parent to every output unit which means the maximal depth is updated :
if $d(n+1) = D, \forall i \in \{n_{I+1}, \ldots, n\}, (n+1, i) \in \pi^0$

Remark 2. In Step 4 we sample the primary parents $j|d(j) < D$. The output neurons are thus the only neurons to have multiple primary parents as they are considered as the only neurons that cannot be primary parents themselves

Remark 3. Step 6 of the algorithm ensures that $d(n+1) = d(j) + 1$ as $\forall i \in \pi(n+1), d(i) \leq d(j)$.

Remark 4. The concept of primary parents in Step 8 allows us to balance between the depth and width of the architecture of the network. When there are still only a small number of units in the network, a primary parent can be picked multiple times. If this is the case, Step 8 allows the network to grow deeper and not only wider.

4 Experimental Results

4.1 Experiments with Small Models

We implement our model in PyTorch and evaluate it on AGNews and MNIST [19].

AGNews : AG News (AG's News Corpus) is a sub-dataset of A. Gulli's corpus of news articles constructed by assembling titles and description fields of articles from the 4 largest classes ("World", "Sports", "Business", "Sci/Tech") of AG's Corpus. The AG News contains 30,000 training and 1,900 test samples per class.

MNIST : The handwritten digit benchmark MNIST is a large collection of handwritten digits. It has a training set of 60,000 examples, and a test set of 10,000 examples.

We compare our model with a fully connected network with the same number of parameters as the final number of parameters of our model (after insertions). Our objective is to study the properties of the final loss surface when adding neurons one by one, all other hyper-parameters being equal (number of parameters, activation functions, weight initialization, accuracy, batch size, SGD algorithm, ...).

In order to study the loss surface, we use the metric proposed in [32], and more precisely the scale invariant measure to compare the flatness of the minima found by our different models. This Hessian-based measure for flatness is invariant to rescaling as the authors use a metric on a quotient manifold structure that captures the rescaling that is natural to the space of parameters of neural networks with positively homogeneous activations.

As done in [32], we train each network architecture and dataset up to 100% of training accuracy with stochastic gradient descent (SGD). We test two different batch sizes of 10 and 100 samples for MNIST, and 1 and 16 samples for AGNews. No pre-trained word embeddings are used for AGNews. For both datasets we use the categorical cross entropy loss. On AGNews, the learning rate is initially 10^{-4} and 10^{-3} on MNIST.

We further test two final model sizes, respectively with 10 and 500 hidden neurons for MNIST, and 10 and 100 hidden neurons for AGNews. On both the AGNews and

MNIST corpora, we train the growing models for 100 epochs, reaching 100% of training accuracy. We let the model learn the new inserted parameters for a few epochs before inserting new neurons again. We used the following insertion scheme :

- (MNIST, 10 final hidden neurons) 1 new unit every 10 epochs from epoch 5
- (AGNews, 10 final hidden neurons) 1 new unit every 10 epochs from epoch 5
- (MNIST, 500 final hidden neurons) 50 new units every 10 epochs from epoch 10 to 60
- (AGNews, 100 final hidden neurons) 10 new units every 10 epochs from epoch 5

The initialization scheme for all the models follows a uniform distribution between -0.5 and 0.5. The inserted neurons are thus initialized in the same way as those in the standard feedforward neural networks we compare our network with. In this way, the insertion process is the only difference between the two training methods, every hyper-parameter being equal otherwise.

Furthermore, we train different feedforward neural network architectures with a various number of hidden layers:

- (MNIST, 10 final hidden neurons) :
  - 1 hidden layer of 10 neurons
- (AGNews, 10 final hidden neurons)
  - 1 hidden layer of 10 neurons
- (MNIST, 500 final hidden neurons)
  - **architecture a** : 1 hidden layer of 500 neurons
  - **architecture b** : 2 hidden layers of 400 then 100 neurons
  - **architecture c** : 3 hidden layers of 300 then 150 then 50 neurons
- (AGNews, 100 final hidden neurons)
  - **architecture a** : 1 hidden layer of 100 neurons
  - **architecture b** : 2 hidden layers of 80 then 20 neurons
  - **architecture c** : 3 hidden layers of 60 then 30 then 10 neurons

Testing multiple architectures with the same number of neurons allows us to verify whether the results presented in Table 1 are due to the insertion process or are influenced by the possible difference in terms of depth between the standard feedforward networks and the layered structure obtained through our growing process.

4.2 Growing RoBERTa's classification head

BERT [6] is a well-known and reference contextual word embeddings model that is pre-trained with a Denoising Autoencoding objective and is at the basis of most state of the art results in many Natural Language Processing (NLP) tasks. RoBERTa ([25]) builds on BERT's language masking strategy, but fine-tunes the original BERT model with a different choice of tasks and conditions.

As most of today's state of the art NLP models, RoBERTa is a complex and large neural network, which thus faces the issue of over-parametrization. It is composed of 355 million parameters stored in 24 layers of self-attention, with a classification head on top to output the desired number of classes. In our experiments, we strictly keep the same

Corpus : MNIST			
Model	**Batch Size**	**Test Accuracy**	**Spectral Norm**
10 hidden neurons			
Growing	10	84%	**2.23**
	100	77.91%	**24.29**
Fully Connected	10	84.03%	38.01
	100	77.74%	68.39
500 hidden neurons			
Growing	10	**96.98%**	**4.45**
	100	**94.84%**	**179.01**
Fully Connected a	10	87.27%	1030.07
	100	66.86%	12111.28
Fully Connected b	10	84.57%	820.54
	100	64.17%	21517.66
Fully Connected c	10	84.26%	1253.72
	100	74.01%	12893.59

(a) Results on MNIST corpus

Corpus : AGNews			
Model	**Batch Size**	**Test Accuracy**	**Spectral Norm**
10 hidden neurons			
Growing	1	90.2%	$\mathbf{9.83 \cdot 10^{-5}}$
	16	88.2%	**3.22**
Fully Connected	1	90.1%	$7.91 \cdot 10^{-4}$
	16	88.3%	20.83
100 hidden neurons			
Growing	1	90.7%	**0.57**
	16	89.2%	**31.31**
Fully Connected a	1	90.5%	3.43
	16	89.0%	69.53
Fully Connected b	1	90.4%	7.02
	16	89.0%	60.89
Fully Connected c	1	90.5%	3.25
	16	88.9%	73.28

(b) Results on AGNews

Table 1: Test Accuracy and Spectral Norm of Hessian at Minima for different trained networks.

number of parameters, without adding more complexity to the model. The classification head is classically composed of a two-layers feed-forward network with 1024 hidden neurons. We thus propose next to replace this classification head with a bare one-layer feed-forward network, and to grow it until it reaches the same number of parameters as the reference classification layer. The rest of the pre-trained Roberta large model, i.e., the self-attention layers, stays the same. We use the standard CoLA benchmark [37] to train and evaluate our growing model against RoBERTa. The CoLA dataset is composed

of 9594 training and 1063 test sentences. The objective is to classify each sentence into two classes: grammatically correct and incorrect English sentences.

In order to study the impact of growing the network, we insert neurons progressively throughout the whole learning process, which consists in 10 epochs in the original RoBERTa experiment. To do so, a total of 1024 hidden neurons are inserted through three different phases: after the second, fifth and seventh epoch. We also compute the sharpness of both the original RoBERTA model and of our grown model, and compare the models performances and loss characteristics. The batch size used is 16. The results are compiled in Table 2.

Model	Test Accuracy	Spectral Norm
Growing	68.2%	$\mathbf{1.8945 \cdot 10^5}$
Fully Connected	68%	$6.7778 \cdot 10^5$

Table 2: Test Accuracy and Spectral Norm of Hessian at Minima on COLA

5 Discussion

The results obtained with our small models experiments first confirm what was shown in [32], i.e., that by increasing the batch size, we increase the sharpness of the minima. But these experiments further show that inserting neurons during the training phase of the model allows to decrease the sharpness of the minima. These results are obtained with shallow networks and standard but relatively simple classification tasks.

Another interesting observation is that our method not only decreases the spectral norm and thus improves the flatness of the minima of a shallow network, but we can also observe in Table 1, that a lower spectral norm seems to correlate with higher generalization performances, as the test accuracies are better for the growing networks when the number of parameters is larger.

Another interesting conclusion that can be drawn from Table 2 is that these good results translate to complex and state-of-the-art neural networks. Indeed, we can observe that growing the final classification head of such a model leads to an optimum solution that is flatter by an order of magnitude, according to the flatness metric, without any negative impact in terms of accuracy. We note that the classification head represents approximately only 0.6% of the total number of parameters of the model, which may explain why the models performances in terms of accuracy are similar. Despite the fact that only a small proportion of the total number of parameters is grown, it is interesting to note that the growing process leads to spectral norm of the optimum that is much lower, which means that we reach a flatter minimum.

Our results tend to show that the growing paradigm, more and more used in the Neural Architecture Search field, is an important asset to reach flatter minima. However, there is no theoretical guarantee for now that a flatter minimum will systematically translate into better generalization performances, although several related works results tend to exhibit such a correlation.

With regard to this question, we can observe in our experiments that with very small neural networks (the case with only 10 hidden neurons in Table 1), the performances are similar although the growing method achieves a minimum that is at least 2.5 (up to more than 10) times flatter than the standard method. We believe that this is due to the small capacity of the neural network, which can not learn more because of its limited size.

Second, when inserting a greater amount of neurons, we can observe two different behaviors. On the one hand, when comparing on MNIST, the growing method leads to flatter minima and better Test Accuracy, as it was also shown in [32]. On the other hand, the flatter minima given by the growing approach do not correlate with better performances on AG News: models with 10 and 100 hidden neurons have roughly the same performances. Similarly, when trained on CoLA with a complex structure, the growing method still leads to a flatter minimum but no significant improvement in accuracy is observed. Our hypothesis is that, in these two cases, the number of parameters of the growing model is too small when compared to the total number of parameters of the model to have a real impact on accuracy, as most of the information is learned in the embeddings.

6 Conclusion and Future Work

Our main contribution is the proposal of a growing neural network approach, which we experimentally validate in several conditions on three tasks and with small and large deep learning models. The results show that the resulting loss function has flatter minima than with the traditional training procedure on the full network. We further show that such flatter minima improves the generalization capability of the trained models when they do not rely on complex embeddings. These results tend to show that the paradigm of growing neural networks during the learning phase intrisically leads to flatter minima, which is an interesting observation, although these results have to be confirmed on different datasets, potentially with transfer learning experiments in order to better assess the generalization performances across related tasks. Furthermore, in the case of Natural Language Processing tasks, a potentially interesting extension could be to adapt the approach to further grow the embeddings in order to have a greater impact on both the loss surface characteristics and model's performances. As this work focuses on feedforward neural networks, an extension to more complex structures, such as convolutions and recurrent networks is also envisaged.

References

1. Andrew R. Barron. Approximation and estimation bounds for artificial neural networks. *Mach. Learn.*, 14(1):115–133, January 1994.
2. Pratik Chaudhari, Anna Choromanska, Stefano Soatto, Yann LeCun, Carlo Baldassi, Christian Borgs, Jennifer Chayes, Levent Sagun, and Riccardo Zecchina. Entropy-SGD: Biasing Gradient Descent Into Wide Valleys. *arXiv e-prints*, page arXiv:1611.01838, November 2016.
3. Anna Choromanska, Mikael Henaff, Michael Mathieu, Gérard Ben Arous, and Yann LeCun. The Loss Surfaces of Multilayer Networks. *arXiv e-prints*, page arXiv:1412.0233, November 2014.

4. Xiaoliang Dai, Hongxu Yin, and Niraj K. Jha. NeST: A Neural Network Synthesis Tool Based on a Grow-and-Prune Paradigm. *arXiv e-prints*, page arXiv:1711.02017, November 2017.
5. Xiaoliang Dai, Hongxu Yin, and Niraj K. Jha. Grow and Prune Compact, Fast, and Accurate LSTMs. *arXiv e-prints*, page arXiv:1805.11797, May 2018.
6. Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *arXiv e-prints*, page arXiv:1810.04805, October 2018.
7. Laurent Dinh, Razvan Pascanu, Samy Bengio, and Yoshua Bengio. Sharp Minima Can Generalize For Deep Nets. *arXiv e-prints*, page arXiv:1703.04933, March 2017.
8. Thomas Elsken, Jan Hendrik Metzen, and Frank Hutter. Neural Architecture Search: A Survey. *arXiv e-prints*, page arXiv:1808.05377, August 2018.
9. Pierre Foret, Ariel Kleiner, Hossein Mobahi, and Behnam Neyshabur. Sharpness-Aware Minimization for Efficiently Improving Generalization. *arXiv e-prints*, page arXiv:2010.01412, October 2020.
10. Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative Adversarial Networks. *arXiv e-prints*, page arXiv:1406.2661, June 2014.
11. Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. Speech Recognition with Deep Recurrent Neural Networks. *arXiv e-prints*, page arXiv:1303.5778, March 2013.
12. P. Healy and Nikola S. Nikolov. How to layer a directed acyclic graph. In *Graph Drawing*, 2001.
13. Steven C. Y. Hung, Cheng-Hao Tu, Cheng-En Wu, Chien-Hung Chen, Yi-Ming Chan, and Chu-Song Chen. Compacting, Picking and Growing for Unforgetting Continual Learning. *arXiv e-prints*, page arXiv:1910.06562, October 2019.
14. Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive Growing of GANs for Improved Quality, Stability, and Variation. *arXiv e-prints*, page arXiv:1710.10196, October 2017.
15. Kenji Kawaguchi. Deep Learning without Poor Local Minima. *arXiv e-prints*, page arXiv:1605.07110, May 2016.
16. Kenji Kawaguchi and Leslie Pack Kaelbling. Elimination of All Bad Local Minima in Deep Learning. *arXiv e-prints*, page arXiv:1901.00279, January 2019.
17. Kenji Kawaguchi and Leslie Pack Kaelbling. Elimination of All Bad Local Minima in Deep Learning. *arXiv e-prints*, page arXiv:1901.00279, January 2019.
18. Kenji Kawaguchi, Leslie Pack Kaelbling, and Yoshua Bengio. Generalization in Deep Learning. *arXiv e-prints*, page arXiv:1710.05468, October 2017.
19. Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.
20. Yann Lecun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature Cell Biology*, 521(7553):436–444, May 2015.
21. Moshe Leshno, Vladimir Ya. Lin, Allan Pinkus, and Shimon Schocken. Multilayer feedforward networks with a nonpolynomial activation function can approximate any function, 1993.
22. Liam Li and Ameet Talwalkar. Random Search and Reproducibility for Neural Architecture Search. *arXiv e-prints*, page arXiv:1902.07638, February 2019.
23. Xilai Li, Yingbo Zhou, Tianfu Wu, Richard Socher, and Caiming Xiong. Learn to Grow: A Continual Structure Learning Framework for Overcoming Catastrophic Forgetting. *arXiv e-prints*, page arXiv:1904.00310, March 2019.
24. Shiyu Liang, Ruoyu Sun, Jason D. Lee, and R. Srikant. Adding One Neuron Can Eliminate All Bad Local Minima. *arXiv e-prints*, page arXiv:1805.08671, May 2018.

25. Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. RoBERTa: A Robustly Optimized BERT Pretraining Approach. *arXiv e-prints*, page arXiv:1907.11692, July 2019.
26. Michael McCloskey and Neal J. Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. *Psychology of Learning and Motivation - Advances in Research and Theory*, 24(C):109–165, January 1989. Funding Information: The research reported in this chapter was supported by NIH grant NS21047 to Michael McCloskey, and by a grant from the Sloan Foundation to Neal Cohen. We thank Sean Purcell and Andrew Olson for assistance in generating the figures, and Alfonso Caramazza, Walter Harley, Paul Macaruso, Jay McClelland, Andrew Olson, Brenda Rapp, Roger Rat-cliff, David Rumelhart, and Terry Sejnowski for helpful discussions.
27. Renato Negrinho, Darshan Patil, Nghia Le, Daniel Ferreira, Matthew Gormley, and Geoffrey Gordon. Towards modular and programmable architecture search. *arXiv e-prints*, page arXiv:1909.13404, September 2019.
28. Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y. Ng. Reading digits in natural images with unsupervised feature learning. In *NIPS Workshop on Deep Learning and Unsupervised Feature Learning 2011*, 2011.
29. Behnam Neyshabur, Ryota Tomioka, and Nathan Srebro. Norm-Based Capacity Control in Neural Networks. *arXiv e-prints*, page arXiv:1503.00036, February 2015.
30. German I. Parisi, Ronald Kemker, Jose L. Part, Christopher Kanan, and Stefan Wermter. Continual Lifelong Learning with Neural Networks: A Review. *arXiv e-prints*, page arXiv:1802.07569, February 2018.
31. Henning Petzka, Linara Adilova, Michael Kamp, and Cristian Sminchisescu. A Reparameterization-Invariant Flatness Measure for Deep Neural Networks. *arXiv e-prints*, page arXiv:1912.00058, November 2019.
32. Akshay Rangamani, Nam H. Nguyen, Abhishek Kumar, Dzung Phan, Sang H. Chin, and Trac D. Tran. A Scale Invariant Flatness Measure for Deep Network Minima. *arXiv e-prints*, page arXiv:1902.02434, February 2019.
33. Nitish Shirish Keskar, Dheevatsa Mudigere, Jorge Nocedal, Mikhail Smelyanskiy, and Ping Tak Peter Tang. On Large-Batch Training for Deep Learning: Generalization Gap and Sharp Minima. *arXiv e-prints*, page arXiv:1609.04836, September 2016.
34. Samarth Sinha, Animesh Garg, and Hugo Larochelle. Curriculum By Smoothing. *arXiv e-prints*, page arXiv:2003.01367, March 2020.
35. Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. Sequence to Sequence Learning with Neural Networks. *arXiv e-prints*, page arXiv:1409.3215, September 2014.
36. Dilin Wang, Meng Li, Lemeng Wu, Vikas Chandra, and Qiang Liu. Energy-Aware Neural Architecture Optimization with Fast Splitting Steepest Descent. *arXiv e-prints*, page arXiv:1910.03103, October 2019.
37. Alex Warstadt, Amanpreet Singh, and Samuel R. Bowman. Neural Network Acceptability Judgments. *arXiv e-prints*, page arXiv:1805.12471, May 2018.
38. Wei Wen, Yandan Wang, Feng Yan, Cong Xu, Chunpeng Wu, Yiran Chen, and Hai Li. SmoothOut: Smoothing Out Sharp Minima to Improve Generalization in Deep Learning. *arXiv e-prints*, page arXiv:1805.07898, May 2018.
39. Lemeng Wu, Bo Liu, Peter Stone, and Qiang Liu. Firefly Neural Architecture Descent: a General Approach for Growing Neural Networks. *arXiv e-prints*, page arXiv:2102.08574, February 2021.