FedLM: A Review on Federated Learning for Language Models

Muhammad Umar Salman
MBZUAI
umar.salman@mbzuai.ac.ae

Abstract

Language Modelling (LM) is one of the most widely used techniques in natural language processing (NLP). These neural language models are pre-trained on large textual corpora to achieve state-of-the-art performances and results. Nowadays, most user generated text come from mobile devices applications but due to strict privacy regulations and concerns, sensitive data cannot be transferred out of the user's device. For this purpose, we look into a federated learning (FL) approach to learn personalized language models that can be deployed on a user's virtual keyboard. FL is a decentralized approach which aims to learn personalized models from multiple distributed edge devices without sharing data to a centralized server. This approach not only addresses the privacy issue but also deals with the limited compute capacity and latency on edge devices. In this survey we discuss different approaches to FL in language modelling over the recent years and will address what each model is trying to achieve along with their strengths and weaknesses. Finally, we will see how most approaches through experimentation outperform server-based training by using federated training instead.

Keywords: Language Modelling, Federated Learning, Natural Language Processing, Neural Language Model

1. Introduction

1.1. Background

Over the last few years revolutionary pre-trained LMs such as BERT [6] have dominated the field of NLP. Most modern applications such as Question Answering, Machine Translation, Relation Extraction, Information Retrieval, Text classification and Text Summarizing etc. all have started to increasingly rely on huge pre-trained models and computationally expensive deep learning algorithms, both of which require large textual training data and a high compute capacity to achieve top performances. LMs are defined as the probability distribution over a sequence of words and recently most textual data has been generated from the mobile platform such as chat apps [2] or surfing over the web. As users have increasingly started to shift to mobile devices [1] there is a need to improve on LMs on the mobile platform. LMs are deployed on virtual mobile keyboards such as Gboard [1] – the Google Keyboard – which works on word-level predictions such as next word prediction, emoji prediction and query suggestion etc. to enrich the user experience. (Figure 1) below displays an example of the next word prediction problem on a virtual keyboard.

Our main contribution in this review paper is discussing different approaches of LMs in FL settings and comparing them to server-based approaches. We also highlight on the strengths and weakness for these approaches and what problem these approaches set out to solve for real world applications.

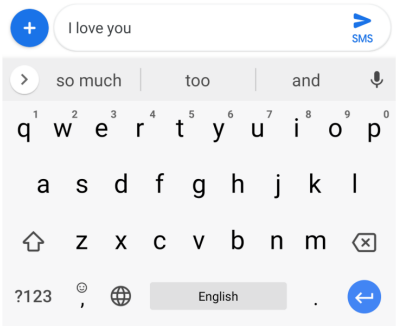

Figure 1. The figure above shows an example of a LM deployed on a virtual keyboard to attempt to predict the next word based on what the user has currently typed. Here 'so much', 'too' and 'and' are all next word prediction given by the LM. Image Source: [11]

1.2. Related Work

Previously Gboard predictions were generated through a word n-gram finite state transducer (FST) [17]. Due to advancements in deep learning language models there has been an increase in RNN-LMs and its variants for word-level prediction tasks. Users expect these language models to make suggestions to an input within 20 milliseconds.

[1] gboard.app.goo.gl/get

However, because of limitations on the device, such as computational resource and low latency, huge language models with millions of parameters are not supported by the mobile platform. State of the art RNN-LMs require over 50MB of memory to store around 25M (million) parameters [24]. To address these bottlenecks for on-device keyboards, different work has been done to lower the model's size and decrease the average prediction time. Compression techniques such as embedded deep learning approach for next word prediction [23] has managed to reduce the model size to under 10MB with a mean prediction time of around 6ms(milliseconds). Although, approaches like these did manage to align their storage and compute capacity with mobile platforms they had a severe disadvantage of sharing user's personal data to a central server. Nowadays, due to rising number of regulation policies regarding data privacy such as General Data Protection Regulation (GDPR) [20], sensitive and personal data collected from end users are not allowed to be transferred to a centralized server.

1.3. Federated Learning

Sharing private data to a central server or other clients has become major concern for many organizations such as hospitals, legal firms and financial institutions. For this reason, we look into LMs which make use of FL as their learning framework. FL allows individual users to train their models while preventing their private local data from leaving their devices to give them a more personalized experience. The process of FL works over a series of rounds where a selected number of participant edge devices individually download a copy of the global model parameters. Using those parameters, the model trains on the users private local data and perform local model updates. These edge devices then upload their model updates which is the difference between the updated parameters and the downloaded parameters. The server then aggregates all the selected client contributions and updates to the global model. This is all done without exposing the user's personal data or letting the data leak out. Common data preservation techniques used throughout our approaches include: differential privacy [7], secure Multi-Party Computation [9], homomorphic encryption [18] and trusted execution environments [8]. (Figure 2) clearly shows the general process that is gone through during a training round of FL.

2. Method

Most approaches [11, 19, 3, 14, 4] in this domain are variants of Long-Short Term Memory models (LSTMs) [12]. As most literature is done primarily on Google's keyboard (Gboard) we can see that most models are trained on Google's personal Gboard dataset and the work is also published by Google LLC authors. In this section we will review the aforementioned approaches and highlight the problems they are solving, techniques they are using and their strengths and weaknesses.

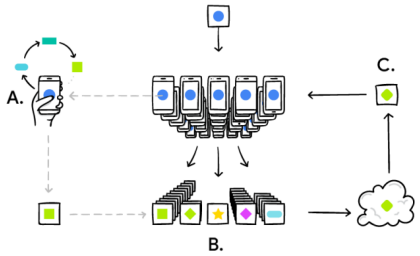

Figure 2. The figure above shows the general federated learning process where in (A) the edge devices compute the updates on local training data after which they upload to (B) where the server uses an aggregating algorithm to aggregate the client updates. After that in (C) new global parameters are sent to the devices for the next round and the process repeats itself. Image Source: [11]

One approach [11] makes use of Coupled Input and Forget Gate (CIFG) [10] for next word predictions while considering the limited compute and storage capacity of mobile devices. The CIFG architecture gives a great advantage to mobile platforms as they, without any loss in performance, manage to significantly reduce the parameter size by 25% as well as lowering the LMs complexity and inference time latency. The CIFG structure comprises of a single gate which controls both the input and recurrent cell self-connections just like the LSTMs special case the Gated Recurrent Unit (GRU) [5]. Through removing peephole connections and tying up input embeddings and output projection matrices this technique reduce the model size along with increasing the accelerated training. This approach uses Federated Averaging algorithm (FedAvg) [16] to aggregate the client updates and uses stochastic gradient descent (SGD) as an optimizer on the server side to train the global model. The strength of the federated LSTM-CIFG approach is it outperforms server trained CIFG models and the n-gram baseline model while maintaining the security and privacy of the client's data. However, this model requires a lot of federated training rounds which leads to high communication costs.

[19] employs transfer learning to pre-train a CIFG network for emoji prediction. The notion of pre-training parameters to improve performance on language modelling tasks can be seen in [13] which proves pre-training gives better results. This particular pre-training process pre-trains all the layers save the final output projection layer. We managed to observe that the emoji model converges faster and gives an enhanced performance particularly because of the sharing of embeddings between the inputs and output. This approach also introduces a triggering model that's purpose is to predict whether the next occurrence should be a word or an emoji. For e.g., user is more likely to add a heart emoji after "I love you" than "I love". Adding a binary classifier

to predict this creates an overhead and for that reason the model incorporates this by assigning all words as a single token and adding it to the unique list of emojis being predicted. This model manages to outperform the pre-trained server-based CIFG model using momentum with Nesterov accelerated gradients which do a much better job in terms of convergence rate and model performance. However, one weakness of this approach is it has been tested on Gboard's dataset whose distribution of emojis shown in (Figure 3) is very light-tailed thus the model more often than not predicts the most frequent emojis and faces the issue of imbalanced classes.

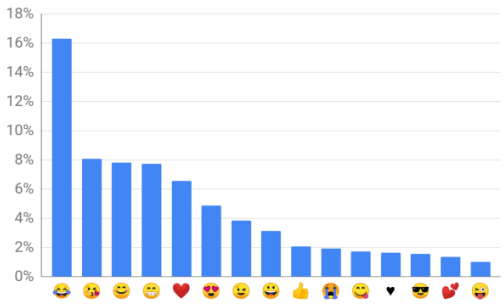

Figure 3. The figure above shows light-tailed distribution of this model which leads to higher prediction of emojis with a higher frequency. Image Source: [19]

[3] uses a character-level RNN [22] to train a federated character-level LSTM-CIFG. They keep peephole connections to learn out-of-vocabulary (OOV) words so it can expand the vocabulary of LMs deployed on virtual keyboards. Since most words typed on a keyboard can either be slang, abbreviations, names, foreign words or typos, models face the issue of including all the words in the vocabulary because of the massive size. For this reason, it is important to include the most frequently typed words so that the LM can predict those frequently used words to the user. So, for enhanced learning representation of such word occurrences, the authors use multiple layers for the LSTM along with a projection layer to reduces the output dimension and accelerates the training. OOVs are then Monte-Carlo sampled on the servers during inference to predict frequent user slang/abbreviation words. The use of character level RNNs and Monte Carlo sampling can be visualized in the (Figure 4). The authors also argue that the use of momentum and adaptive L2-norm clipping on client updates helps with the robustness of the model convergence. The strength of the paper is incorporating unknown frequently used words into next word predictions. However, a weakness of this approach would be the frequent unintended typos which may pop-up for the user in next word suggestions.

We can see that most works have used FedAvg as a client updates aggregator. [14] introduces an attention mechanism

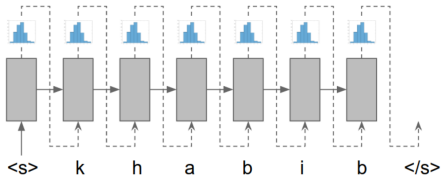

Figure 4. The figure above shows character-based RNN using Monte-Carlo sampling which attempt to solve OOV problems such as typos, slang and abbreviations etc. Image Source: [3]

for model aggregation of client updates. This particular optimization algorithm is termed as Attentive Federated Aggregation (FedAtt). The model leverages GRU which is a simpler variant of LSTM and like CIFG also manages to reduce model complexity on parameter space without dropping performance levels. The FedAtt through local quick adaptations on the client training manages to get a good generalization which saves on communication rounds by reducing the training rounds thus accelerating the learning process. This generalization is determined by the relative importance of each client to the server. This mechanism automatically assigns weights between the server and client edge devices based on the similarity between the two. A major strength for this approach is that with fewer communication rounds this technique outperforms FedAvg algorithm in most settings.

[4] mentions that the simple use of RNN may not be the way to go about deploying LMs due to the large parameters. They claim that it leads to slow inference and for this reason, they propose a n-gram LM which is generated from a federated RNN. This approach overcomes considerable memory problems while still managing to keep high levels of performances by the introduction of an approximation algorithm whose concepts are based off of sample approximation [21]. It also attempts to tackle the capitalization issue in which the model keeps extra memory to store which words are capitalized and which aren't. Like many other approaches this one also makes use of CIFG but in addition to that it also employs group-LSTM (GLSTM) [15]. The model is first trained on uncased user data and then using sample approximation is then tuned to case those words from the uncased LM. The strength of the paper lies in reducing the large memory footprint that was being used for capitalization of words. The weakness on the other hand is the overhead of computation for fine tuning the approximation.

3. Results and Discussion

In this section we will look at the experiments conducted by each of the papers, what they compare their model against and how well each of them performed. Most of the

Paper Ref.	Problem	Dataset	Aggregator	Model	Optimizer
[11]	Next Word Prediction	Gboard	FedAvg	LSTM with CIFG	SGD
[19]	Emoji Prediction	Gboard	FedAvg	Pretrained LSTM with CIFG	Nesterov Accelerated Gradient
[3]	OOVs Prediction	Gboard, Reddit	FedAvg	LSTM with CIFG	SGD with Momentum
[14]	Federated Attention Algorithm	PennTreebank, Reddit, WikiText	FedAtt	GRU	SGD with Momentum
[4]	Capitalization Prdeiction	Gboard	FedAvg	LSTM with CIFG + GLSTM	SGD

Table 1. The table above shows the broad overview of all the approaches we have seen specifically the problem they are trying to solve and the basic features each approach uses to help them achieve this.

approaches by each paper were not trying to do better than each other which is why it is hard to compare all the approaches proposed based on a single metric. Each proposed approach attempts to solve a separate issue thus having its own strengths and weaknesses in the realm of LM in federated learning settings. For this purpose, we have put together in (Table 1) which includes the most basic features of the papers to see the major differentiating factors for each of them.

One thing we can gather from this table is [14] which uses FedAtt as its aggregator algorithm to amalgamate all the client updates rather than FedAvg. (Figure 5) helps to give a general idea about aggregators used in FL settings.

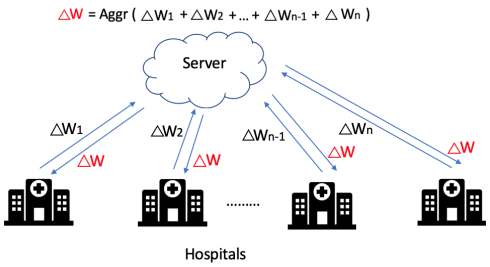

Figure 5. The figure above shows the flow of parameters and the aggregator algorithm which sends global model parameters from server to clients and then the selected clients return their updated parameters trained on their local data back to the server which applies the aggregator function on the client updates

The main difference between both algorithms is that FedAvg on receiving updates from clients simply averages it out whereas in FedAtt we look at the relative importance assigns weights to the client based on the similarity thus allowing it to generalize well. This generalization leads to fewer training rounds thus fewer communication rounds which is preferred on a mobile platform. [14] conducts its experiments on multiple datasets which can be seen in (Table 2). It shows the number of examples both models were trained and tested on. These include Penn Treebank [2], WikiText-2 [3] and Reddit comments [4]. These experiments highlight the effect of both aggregators in a FL environment.

[2]Penn Treebank:https://github.com/wojzaremba/lstm/tree/master/data

[3]WikiText-2:https://s3.amazonaws.com/research.metamind.io/wikitext/wikitext-2-v1.zip

[4]Reddit Comments:https://www.kaggle.com/kaggle/reddit-comments-may-2015

Dataset	# Train	# Valid	#Test
Penn Treebank	887,521	217,646	245,569
WikiText-2	2,088,628	217,646	245,569
Reddit Comments	1,784,023	102,862	97,940

Table 2. Table above shows the number of examples used for each dataset and in terms of training, validation and testing examples. Table Source: [14]

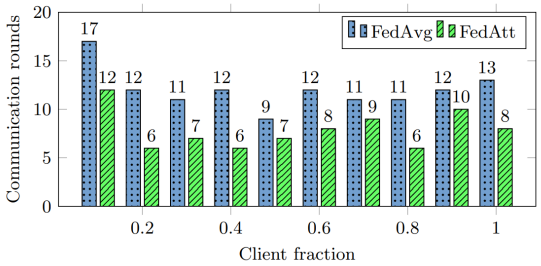

Figure 6. The figure shows the communication rounds used in FedAvg and FedAtt for different number of clients. Where FedAtt always has fewer number of communication rounds. Image Source: [14]

From (Figure 6) we can see that under similar settings and similar data that FedAtt manages to use fewer communication rounds for training for different number of clients with the same level of accuracy as FedAvg. [14] thus does well to show why FedAtt should be used instead of FedAvg in cases where there are communication constraints.

We also see that most of the work is presented by authors from Google LLC which is why they make use of the Gboard dataset. Most of their work [11, 19, 3, 4] are trying to solve different problems for example [11] attempts to reduce the number of parameters so as to decrease the memory size of the LM deployed on the virtual keyboard while giving better performance than server-base solutions. This is seen to have been achieved in their experiments and in the (Table 3) where they outperform both baseline n-gram models as well as server trained CIFG models in top 1% and 3% recall scores. However, this method does have a drawback as it requires a large amount of training rounds subsequently higher rounds for communication on the mobile platform.

In [19] we see the authors trying to solve the prediction of emoji's in FL using a pretrained LSTM-CIFG. Here they

Model	Top-1 Recall[%]	Top-3 Recall[%]
N-gram	5.24 ± 0.02	11.05 ± 0.03
Server CIFG	5.76 ± 0.03	13.63 ± 0.04
Federated CIFG	5.82 ± 0.03	13.75 ± 0.03

Table 3. The figure above shows the results of experiments obtained in [11] where it clearly shows that the federated learning approach has performed slightly better than server-based learning in to 1% and 3% recall

not only have to determine which emoji has to be predicted but also whether a word should be suggested or whether an emoji which the authors smartly incorporate within the model to avoid any overhead of using another binary classifier model to solve that problem. In (Table 4) we see that the proposed model in comparison to server trained settings matches if not exceeds accuracies under FL settings. The only drawback of this model is the distribution of emojis. Few emojis have a very high frequency leading to higher predictions being from those limited emojis.

Model	Accuracy (1 round)	AUC
Best Federated	0.256	0.863
Best Server Trained	0.239	0.898

Table 4. The figure above shows the results of experiments obtained in [19] where it shows that the best federated learning achieves better results than server-based learning in terms of accuracy

In [3, 4] we again see two unique problems trying to be solved. In [3] we see the model trying to distinguish between frequently used slang which should be used for prediction in the future versus frequent typos which one would not want be predicted in the next word suggestions. The authors conduct an experiment where they simulate Reddit data to find probabilities of the OOV words and compare it to the ground truth probabilities that came from the Reddit data. The top 10 OOV words and their probabilities have been shown in (Figure 7). In [4] we see the model is trying to solve the large memory issue posed by learning capitals which we it does very well using sample approximation. This approach like many other is shown to perform better than baseline server-based learning which they show through A/B experiments done on real time users.

Looking at the results from different experiments we can see that all the approaches using FL framework perform well against centralized learning frameworks as in most cases not only match up to the centralized training performance but exceed their accuracies and results. Each approach attempts in addition to reducing parameter size and communication rounds also tries to solve a new problem using innovative solutions. However, all the solutions we have reviewed all use some variation of LSTMs or RNNs to build

yea	0.0050	yea	0.0057
upvote	0.0033	upvote	0.0040
downvoted	0.0030	downvoted	0.0033
alot	0.0026	alot	0.0029
downvote	0.0023	downvote	0.0026
downvotes	0.0018	downvotes	0.0022
upvotes	0.0016	upvotes	0.0021
wp-content	0.0016	op's	0.0019
op's	0.0015	wp-content	0.0017
restrict_sr	0.0014	redditors	0.0016

Figure 7. Top 10 out-of-vocabulary words and probability, where on the left is the Reddit dataset ground truth and on the right side is the probabilities generated from simulating the Reddit dataset using FL. Data Source: [3]

LMs. Given the hype and rise of Transformer models in NLP, we expected to see a few approaches using pre-trained transformer-based models on the server side and reduce the dimensions and fine-tune it on local data. Most approaches also use several standard privacy and security techniques and algorithms to keep the preservation of local user data. With the introduction of new advanced threats including reconstruction of weight parameters into original data there is a urgent need of more advanced and secure privacy algorithms to keep users' data safe for LM training. These two main ideas serve as open problems which are worth exploring as future potential research.

4. Conclusion

In this paper, we review different approaches for language modelling using FL. We notice that most approaches compare themselves with server-based results and even though a centralized learning is more likely to yield better results we see that under some FL settings, federated approaches do manage to outperform them. Most of the approaches we review look to solve various problems within the federated learning framework such as emoji prediction, important OOVs prediction etc. They also successfully reduce the parameter sizes and number of communication rounds while avoiding exposing private data of users. This can be seen through multiple experimentation results which show no drop in performance and accuracy between server-based and federated-based methods. However, given the recent success of transformer-based LMs in other NLP tasks, we couldn't find any work relevant to federated learning tasks as most of the approaches we reviewed were all variants of LSTMs, this opens up a huge gap for potential future research work. Another area lacking significant work is the use of advanced privacy and data preserving techniques for the security of a user's data on the virtual keyboard platform. Here too we expect to see more work in the future.

References

[1] Monica Anderson. Technology device ownership: 2015. 2015. 1

[2] Agathe Battestini, Vidya Setlur, and Timothy Sohn. A large scale study of text-messaging use. In *Proceedings of the 12th international conference on Human computer interaction with mobile devices and services*, pages 229–238, 2010. 1

[3] Mingqing Chen, Rajiv Mathews, Tom Ouyang, and Françoise Beaufays. Federated learning of out-of-vocabulary words. *arXiv preprint arXiv:1903.10635*, 2019. 2, 3, 4, 5

[4] Mingqing Chen, Ananda Theertha Suresh, Rajiv Mathews, Adeline Wong, Cyril Allauzen, Françoise Beaufays, and Michael Riley. Federated learning of n-gram language models. *arXiv preprint arXiv:1910.03432*, 2019. 2, 3, 4, 5

[5] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014. 2

[6] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018. 1

[7] Cynthia Dwork. Differential privacy: A survey of results. In *International conference on theory and applications of models of computation*, pages 1–19. Springer, 2008. 2

[8] Jan-Erik Ekberg, Kari Kostiainen, and N Asokan. Trusted execution environments on mobile devices. In *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*, pages 1497–1498, 2013. 2

[9] Oded Goldreich. Secure multi-party computation. *Manuscript. Preliminary version*, 78, 1998. 2

[10] Klaus Greff, Rupesh K Srivastava, Jan Koutník, Bas R Steunebrink, and Jürgen Schmidhuber. Lstm: A search space odyssey. *IEEE transactions on neural networks and learning systems*, 28(10):2222–2232, 2016. 2

[11] Andrew Hard, Kanishka Rao, Rajiv Mathews, Swaroop Ramaswamy, Françoise Beaufays, Sean Augenstein, Hubert Eichner, Chloé Kiddon, and Daniel Ramage. Federated learning for mobile keyboard prediction. *arXiv preprint arXiv:1811.03604*, 2018. 1, 2, 4, 5

[12] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997. 2

[13] Jeremy Howard and Sebastian Ruder. Universal language model fine-tuning for text classification. *arXiv preprint arXiv:1801.06146*, 2018. 2

[14] Shaoxiong Ji, Shirui Pan, Guodong Long, Xue Li, Jing Jiang, and Zi Huang. Learning private neural language modeling with attentive aggregation. In *2019 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2019. 2, 3, 4

[15] Oleksii Kuchaiev and Boris Ginsburg. Factorization tricks for lstm networks. *arXiv preprint arXiv:1703.10722*, 2017. 3

[16] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*, pages 1273–1282. PMLR, 2017. 2

[17] Mehryar Mohri. Finite-state transducers in language and speech processing. *Computational linguistics*, 23(2):269–311, 1997. 1

[18] Michael Naehrig, Kristin Lauter, and Vinod Vaikuntanathan. Can homomorphic encryption be practical? In *Proceedings of the 3rd ACM workshop on Cloud computing security workshop*, pages 113–124, 2011. 2

[19] Swaroop Ramaswamy, Rajiv Mathews, Kanishka Rao, and Françoise Beaufays. Federated learning for emoji prediction in a mobile keyboard. *arXiv preprint arXiv:1906.04329*, 2019. 2, 3, 4, 5

[20] General Data Protection Regulation. Regulation eu 2016/679 of the european parliament and of the council of 27 april 2016. *Official Journal of the European Union*, 2016. 2

[21] Ananda Theertha Suresh, Brian Roark, Michael Riley, and Vlad Schogol. Approximating probabilistic models as weighted finite automata. *Computational Linguistics*, 47(2):221–254, 2021. 3

[22] Ronald J Williams and David Zipser. A learning algorithm for continually running fully recurrent neural networks. *Neural computation*, 1(2):270–280, 1989. 3

[23] Seunghak Yu, Nilesh Kulkarni, Haejun Lee, and Jihie Kim. An embedded deep learning based word prediction. *arXiv preprint arXiv:1707.01662*, 2017. 2

[24] Barret Zoph and Quoc V Le. Neural architecture search with reinforcement learning. *arXiv preprint arXiv:1611.01578*, 2016. 2