Homework 3 - Version 1.0

Deadline: Friday, Mar.11, at 11:59pm.

Submission: You must submit your solutions as a PDF file through MarkUs[1]. You can produce the file however you like (e.g. LaTeX, Microsoft Word, scanner), as long as it is readable.

See the syllabus on the course website[2] for detailed policies. You may ask questions about the assignment on Piazza[3] with the tag `hw3`. *Note that 10% of the homework mark (worth 1 pt) may be removed for a lack of neatness.*

The teaching assistants for this assignment are Sheng Jia and Denny Wu.
`mailto:csc413-2022-01-tas@cs.toronto.edu`

[1] `https://markus.teach.cs.toronto.edu/2022-01/courses/16/`
[2] `https://uoft-csc413.github.io/2022/assets/misc/syllabus.pdf`
[3] `https://piazza.com/utoronto.ca/winter2022/csc4132516/`

1 Trading off Resources in Neural Net Training

1.1 Effect of batch size

When training neural networks, it is important to select appropriate learning hyperparameters such as learning rate, batch size, and the optimizer for training. In this question, we will investigate the effect of how varying these quantities will affect the validation loss during training.

1.1.1 Batch size vs. learning rate

Batch size affects the stochasticity in optimization, and therefore affects the choice of learning rate. We demonstrate this via a simple model called the Noisy Quadratic Model (NQM). Despite its simplicity, the NQM captures many essential features in realistic neural network training [Zhang et al., 2019].

For simplicity, we only consider the scalar version of the NQM. We have the quadratic loss $\mathcal{L}(w) = \frac{1}{2}aw^2$, where $a > 0$ and $w \in \mathbb{R}$ is the weight that we would like to optimize. Assume that we only have access to a noisy version of the gradient — each time when we make a query for the gradient, we obtain $g(w)$, which is the true gradient $\nabla\mathcal{L}(w)$ with additive Guassian noise:

$$g(w) = \nabla\mathcal{L}(w) + \epsilon, \quad \epsilon \sim \mathcal{N}(0, \sigma^2).$$
$$\text{SGD updates:} \quad w \leftarrow w - \eta g(w).$$

One way to reduce noise in the gradient is to use minibatch training. Let B be the batch size, and denote the minibatch gradient as $g_B(w)$:

$$g_B(w) = \frac{1}{B}\sum_{i=1}^{B} g_i(w), \quad \text{where } g_i(w) = \nabla\mathcal{L}(w) + \epsilon_i, \quad \epsilon_i \overset{i.i.d.}{\sim} \mathcal{N}(0, \sigma^2).$$
$$\text{Mini-batch SGD updates:} \quad w \leftarrow w - \eta g_B(w).$$

[0.5pt] As batch size increases, how do you expect the optimal learning rate to change? Briefly explain in 2-3 sentences.

(Hint: Think about how the minibatch gradient noise change with B.)

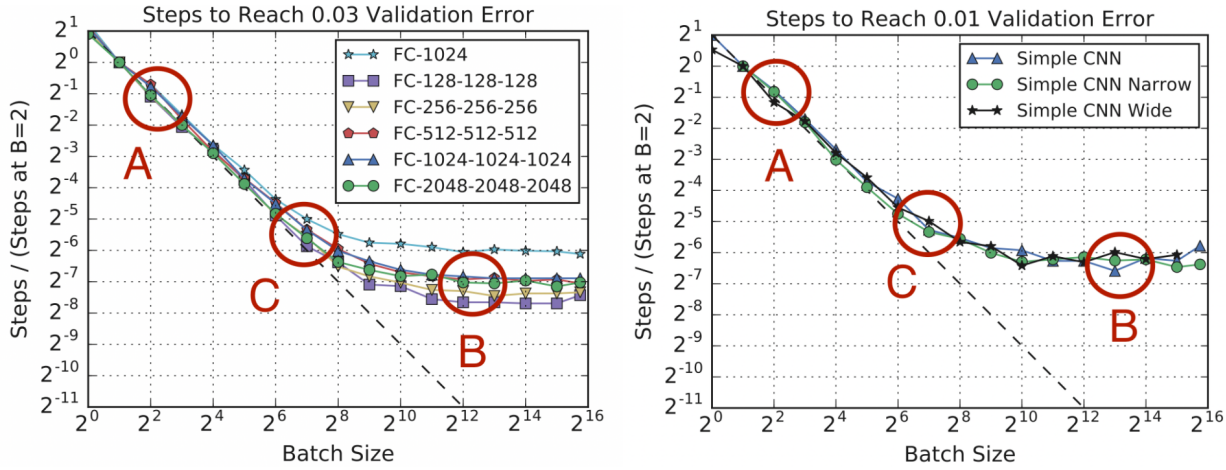

Figure 1: Illustrations of the typical relationship between training steps and the batch size for reaching a certain validation loss under various architectures (from [Shallue et al., 2018]). **Left**: fully-connected models with various sizes. **Right**: Convolutional models with different widths. Learning rate and other related hyperparameters are tuned for each point on the curve.

1.1.2 Training steps vs. batch size

For most of neural network training in the real-world applications, we often observe the relationship of training steps and batch size for reaching a certain validation loss as illustrated in Figure 1. Such a relationship can be observed across various architectures.

(a) **[0.5pt]** For the three points (A, B, C) on Figure 1, which one has the most efficient batch size (in terms of best resource and training time trade-off)? Assume that you have access to scalable (but not free) compute such that minibatches are parallelized efficiently. Briefly explain in 1-2 sentences.

(b) **[0.5pt]** Figure 1 demonstrates that there are often two regimes in neural network training: the noise dominated regime and the curvature dominated regime. In the noise dominated regime, the bottleneck for optimization is that there exists a large amount of gradient noise. In the curvature dominated regime, the bottleneck of optimization is the ill-conditioned loss landscape. For points A and B on Figure 1, which regimes do they belong to? Fill each of the blanks with **one** best suited option. **Options: noise dominated / curvature dominated**.

Point A: Regime: _____. **Point B:** Regime: _____.

1.1.3 Batch size, Optimizer, Normalization, Learning Rate

In this part, we will develop further intuitions about the different optimization regimes. When we encounter optimization difficulties in minimizing a training loss, there are several hyperparameters we can tune to accelerate the training **non-trivially**. In contrast to Figure 1, we now focus on one typical training curve Figure 2 Left where the x-axis represents the number of epochs trained. It is a training curve from ResNet paper [He et al., 2016], and **A is the curvature dominated regime and B is the noise dominated regime.** These terms are explained in the previous question. Note that the learning rate schedule plot is also shown for completeness and explaining the phenomenons that could happen in the given training curve. However, you should not worry

about it when answering this question since your goal is to simply pick the solutions to curvature dominated regime and noise dominated regime from following options.

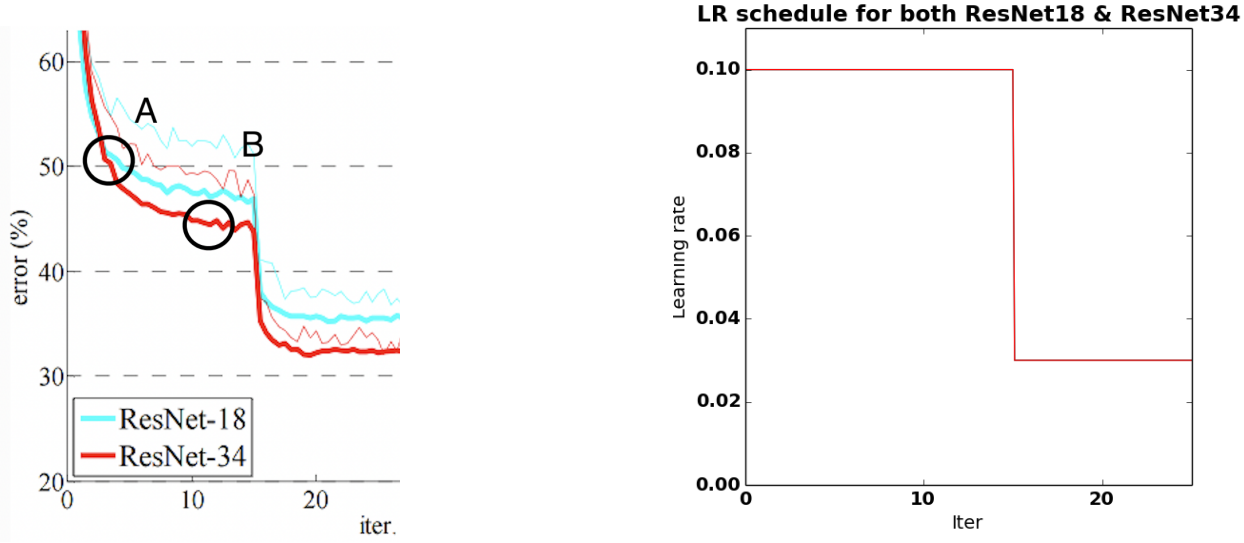

Figure 2: **Left:** The cyan curve shows the training curve of ResNet-18 on Imagenet. The red curve trains ResNet-34. [He et al., 2016] Regime A is dominated by the ill-conditioned landscape. Regime B is dominated by the large amount of gradient noise. **Right:** Corresponding learning rate schedule used for training ResNet.

I Use an adaptive, 2nd, or higher order optimizer

II (II+) Increase the batch size, (II-) Decrease the batch size

III (III+) Increase the learning rate, (III-) Decrease the learning rate

IV Use a normalization technique such as batch norm

(a) **[1pt]** Regime A is dominated by curvature in Figure 2. Which of the above options will accelerate the training by addressing the ill-conditioned curvature.
(Select all that apply from I, II+, II-, III+, III-, IV)

(b) **[1pt]** Regime B is dominated by the noise in Figure 2. Which of the above options will accelerate the training by addressing the large amount of gradient noise?
(Select all that apply from I, II+, II-, III+, III-, IV)

1.2 Model size, dataset size and compute

We have seen in the previous section that batch size and learning rate are important hyperparameter to help with optimization. Besides efficiently minimizing the training loss, we are also interested in the test loss. Recently, researchers have observed an intriguing relationship between the test loss and hyperparameters such as the model size, dataset size and the amount of compute used.

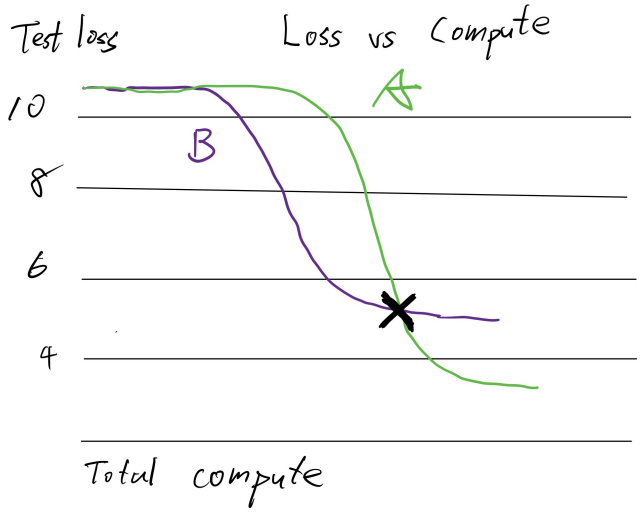

Figure 3: Test loss of language models of different sizes, plotted against the amount of compute (in petaflop/s-days, similar to Fig 2 in [Kaplan et al., 2020]). You can think of the total compute used, shown in x-axis, as *number of training steps* $\times$ *number of parameters* $\times$ *constants*.

[Kaplan et al., 2020] argues for a "Pareto frontier" of optimal allocation of compute, in terms of model size, training steps, to reach a target test loss.

We explore this relationship for neural language models in this section. A similar phenomenon has also been observed in computer vision models. In the same spirit as [Kaplan et al., 2020], we plot the test loss of two models in Figure 3. It compares the test loss vs compute of the trained neural language models with the different number of parameters and training steps.
Note: You can think of the total compute used, shown in x-axis, as *number of training steps* $\times$ *number of parameters* $\times$ *constants*.

(a) **[1pt]** There are two language models shown in Figure 3: Model A in green and Model B in purple. We also highlight an intersection point "X" that denotes a target test loss achieved by both models using the same total amount of compute. (1) Which of the two models has more parameters? Why? (2) At the intersection "X", which of the two models has been training for more iterations/updates? Why?

Explain your answers in no more than 3 sentences.

(b) **[1pt]** Suppose you want to train a language model to reach a desired test loss.

In what situation, will you choose training one over the other (big/small model) if either can give you the same desired test loss using the same total compute? Give a brief explanation. For example, if you have an urgent deadline coming up, which one will require more wallclock time to reach "X"? You can think of the total compute used, shown in x-axis, as *number of training steps* $\times$ *number of parameters* $\times$ *constants*.

2 Generalization Error of Linear Regression

Today's deep learning practices may seem more like alchemy than science, but, we can still obtain useful insight about deep neural networks by studying their linear counterparts. In Homework 1,

we asked you to derive the training loss of a noisy regression problem. Here we will learn to derive the exact test loss, so we can concretely reason about the model's generalization performance in terms of the dataset size and the model size. Many recent works has shown empirically modern deep learning models behave similarly to linear models under gradient descent [Lee et al., 2019].

Recall the linear regression model from homework 1:

$$\hat{\mathbf{w}} = \underset{\mathbf{w}\in\mathbb{R}^d}{\operatorname{argmin}} \frac{1}{n}\sum_{i=1}^{n}(\mathbf{w}^\top \mathbf{x}_i - t_i)^2. \tag{2.1}$$

Denote $\mathbf{X} = [\mathbf{x}_1; \mathbf{x}_2; ...\mathbf{x}_n] \in \mathbb{R}^{n\times d}$ as the training data matrix (design matrix), and $\mathbf{t} \in \mathbb{R}^n$ as the corresponding label vector. When $\mathbf{X}$ has full-rank, the solution of the above problem is given as

$$\hat{\mathbf{w}} = \begin{cases} (\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{X}^\top\mathbf{t}, & n > d, \\ \mathbf{X}^\top(\mathbf{X}\mathbf{X}^\top)^{-1}\mathbf{t}, & n < d. \end{cases} \tag{2.2}$$

We restrict ourselves to a linear teacher model with Gaussian features, i.e.,

$$t_i = \mathbf{w}_*^\top \mathbf{x}_i + \varepsilon_i, \quad \mathbf{x}_i \overset{i.i.d.}{\sim} \mathcal{N}(0, \mathbf{I}_d), \tag{2.3}$$

where ε_i is i.i.d. label noise with mean 0 and variance σ^2, and $\mathbf{w}_* \in \mathbb{R}^d$ is the true signal that does not depend on $\mathbf{x}, \varepsilon$. To further simplify the computation, we consider a Bayesian setting and place an isotropic prior on the true signal[4]: $\mathbb{E}[\mathbf{w}_*\mathbf{w}_*^\top] = \frac{1}{d}\mathbf{I}_d$.

Our goal is to compute the generalization error (test loss) of the linear regression model:

$$\mathcal{R}(\hat{\mathbf{w}}) = \mathbb{E}_{\tilde{\mathbf{x}},\varepsilon,\mathbf{w}_*}(\mathbf{w}_*^\top\tilde{\mathbf{x}} - \hat{\mathbf{w}}^\top\tilde{\mathbf{x}})^2, \tag{2.4}$$

where the test data is drawn from the same distribution: $\tilde{\mathbf{x}} \sim \mathcal{N}(0, \mathbf{I}_d)$.

2.1 Bias-variance decomposition

(Review the concept on Bias-Variance Decomposition from CSC311 `https://www.cs.toronto.edu/~rgrosse/courses/csc311_f21/`) We separate the generalization error into two terms: the bias term, which measures how well we are learning the true parameters $\mathbf{w}_*$, and the variance term, which is due to "overfitting" the noise in the labels ε. For the following two questions, expand the square in (2.4) and simplify the expressions using the expectation over $\tilde{\mathbf{x}}, \varepsilon, \mathbf{w}_*$.

2.1.1 [0.5pt]

When $n > d$ (underparameterized regime), show that the test loss can be written as

$$\mathcal{R}(\hat{\mathbf{w}}) = \underbrace{0}_{\text{bias, } \mathcal{B}(\hat{\mathbf{w}})} + \underbrace{\sigma^2 \operatorname{Tr}\Big((\mathbf{X}^\top\mathbf{X})^{-1}\Big)}_{\text{variance, } \mathcal{V}(\hat{\mathbf{w}})}. \tag{2.5}$$

Hint: use the i.i.d. property of the training data and label noise: $\mathbb{E}[\mathbf{x}\mathbf{x}^\top] = \mathbf{I}_d$, $\mathbb{E}[\boldsymbol{\varepsilon}\boldsymbol{\varepsilon}^\top] = \sigma^2\mathbf{I}_n$, *where* $\boldsymbol{\varepsilon} \in \mathbb{R}^n, [\boldsymbol{\varepsilon}]_i = \varepsilon_i$.

[4]This does not mean that $\mathbf{w}_*$ is random at training time – we take this expectation for the test loss.

2.1.2 [0pt]

When $n < d$ (overparameterized regime), show that the test loss can be written as

$$\mathcal{R}(\hat{\mathbf{w}}) = \underbrace{\frac{1}{d}\operatorname{Tr}\Big(\mathbf{I}_d - \mathbf{X}^\top(\mathbf{X}\mathbf{X}^\top)^{-1}\mathbf{X}\Big)}_{\text{bias, } \mathcal{B}(\hat{\mathbf{w}})} + \underbrace{\sigma^2 \operatorname{Tr}\Big((\mathbf{X}\mathbf{X}^\top)^{-1}\Big)}_{\text{variance, } \mathcal{V}(\hat{\mathbf{w}})}. \tag{2.6}$$

Hint: for the bias term, use the isotropic prior: $\mathbb{E}[\mathbf{w}_*\mathbf{w}_*^\top] = \frac{1}{d}\mathbf{I}_d$.

2.2 Deriving the exact expressions

To derive the precise expressions of the error, we take expectation over the design matrix $\mathbf{X}$ (denote the expected risk as $\mathbb{E}[\mathcal{R}(\hat{\mathbf{w}})]$). You may use the following facts:

- $\operatorname{Tr}\big(\mathbf{I}_d - \mathbf{X}^\top(\mathbf{X}\mathbf{X}^\top)^{-1}\mathbf{X}\big) = n - d$.
- Given Gaussian random matrix $\mathbf{G} \in \mathbb{R}^{m\times p}$, where $[\mathbf{G}]_{ij} \sim \mathcal{N}(0,1)$, we have

$$\mathbb{E}\operatorname{Tr}\Big((\mathbf{G}^\top\mathbf{G})^{-1}\Big) = \frac{p}{m-p-1}, \quad \text{if } m > p+1. \tag{2.7}$$

  Bonus: argue why this is the case, using properties of the χ^2*-distribution.*

2.2.1 [1pt]

Simplify the expression of $\mathbb{E}[\mathcal{R}(\hat{\mathbf{w}})]$ for both the under- and over-parameterized case. Your final result should only involve n, d, σ, and you may ignore the regime $n - 1 \le d \le n + 1$.

2.2.2 Double descent [1pt]

Based on the formulas, (1) under what conditions (on n, d, σ) is it possible for the model to achieve perfect generalization, i.e., $\mathcal{R}(\hat{\mathbf{w}}) = 0$? (2) Does adding more training examples always help generalization? Why? Briefly explain your answers in no more than 3 sentences.

2.2.3 [0pt]

Plot the expression you derived in the previous question with the following parameters: fix the number of features $d = 500$, and vary training set size n from 100 to 1000 (make sure you exclude the regime $n - 1 \le d \le n + 1$). Include your figures for $\sigma = 0$ (noiseless) and $\sigma = 0.2$ (noisy). Answer the following questions based on the figure:

- What difference do you observe between the noiseless and noisy setting?
- Does adding more training data (increasing n) always lead to better test performance? If not, under what conditions (on n, d, σ) is it beneficial? And when does it hurt?

2.3 Ridge regularization

To reduce overfitting to the label noise, we now consider regularizing the ℓ_2-norm of the parameters. For $\lambda > 0$, the regularized objective is given as

$$\hat{\mathbf{w}}_\lambda = \operatorname*{argmin}_{\mathbf{w}\in\mathbb{R}^d} \frac{1}{n}\sum_{i=1}^{n}(\mathbf{w}^\top\mathbf{x}_i - t_i)^2 + \lambda\|\mathbf{w}\|_2^2, \tag{2.8}$$

2.3.1 [0pt]

By taking the derivative of the objective (2.8) and setting it to zero[5], show that $\hat{\mathbf{w}}_\lambda$ has the following closed-form:

$$\hat{\mathbf{w}}_\lambda = (\mathbf{X}^\top \mathbf{X} + n\lambda \mathbf{I}_d)^{-1} \mathbf{X}^\top \mathbf{t}. \tag{2.9}$$

2.3.2 [0.5pt]

Heuristically argue whether the regularization strength λ should increase or decrease with the training set size n and noise level σ. Provide your arguments in no more than two sentences.

2.3.3 [0pt]

Define $\gamma := \frac{d}{n}$ and set $\lambda = \sigma^2 \gamma$ (why does this choice of λ make sense?). Under the same setting as Section 2.1, show that the generalization error (2.4) can be simplified to

$$\mathcal{R}(\hat{\mathbf{w}}_\lambda) = \sigma^2 \operatorname{Tr}\Big((\mathbf{X}^\top \mathbf{X} + n\lambda \mathbf{I}_d)^{-1}\Big). \tag{2.10}$$

2.3.4 [0.5pt]

Recall that $\gamma = \frac{d}{n}$. Due to properties of Gaussian random matrix (in particular, the Stieltjes transform of the Marchenko-Pastur law, e.g., see [Bai and Silverstein, , Section 3]), we know that for $\lambda > 0$,

$$\mathbb{E}[\mathcal{R}(\hat{\mathbf{w}}_\lambda)] = \sigma^2 \mathbb{E}\left[\operatorname{Tr}\Big((\mathbf{X}^\top \mathbf{X} + n\lambda \mathbf{I}_d)^{-1}\Big)\right] = \frac{-(1-\gamma+\lambda) + \sqrt{(1-\gamma+\lambda)^2 + 4\gamma\lambda}}{2\lambda}, \tag{2.11}$$

Using this result, we can plot the expected risk $\mathbb{E}\mathcal{R}(\hat{\mathbf{w}}_\lambda)$ for our choice of $\lambda = \frac{\sigma^2 d}{n} = \sigma^2 \gamma$.

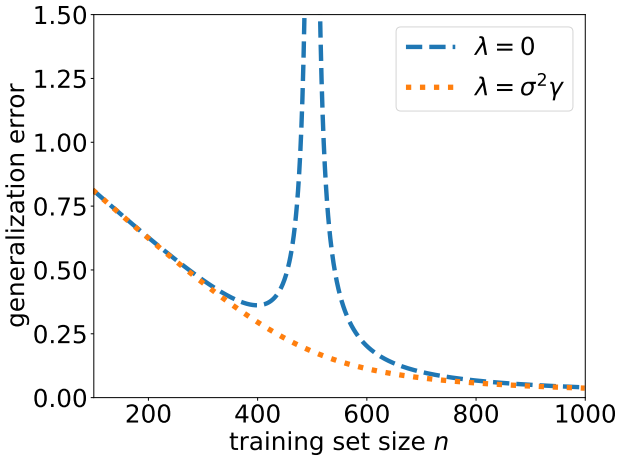

Figure 4: Generalization error of ridge regression. Similar to Section 2.2.3, we fix the number of features $d = 500$, and vary training set size n from 100 to 1000. We choose $\sigma = 0.2$.

Based on the figure, answer the following questions:

[5] We can do this because the objective is strongly convex.

- Compare the test loss of the unregularized estimator $\mathbb{E}[\mathcal{R}(\hat{\mathbf{w}})]$ in Section 2.2.1 against the ridge-regularized $\mathbb{E}[\mathcal{R}(\hat{\mathbf{w}}_\lambda)]$. What difference do you observe?
- Under appropriate ridge regularization ($\lambda = \sigma^2\gamma$), does adding more training data (increasing n) always lead to better test performance?

References

[Bai and Silverstein,] Bai, Z. and Silverstein, J. W. *Spectral analysis of large dimensional random matrices*, volume 20. Springer.

[He et al., 2016] He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

[Kaplan et al., 2020] Kaplan, J., McCandlish, S., Henighan, T., Brown, T. B., Chess, B., Child, R., Gray, S., Radford, A., Wu, J., and Amodei, D. (2020). Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*.

[Lee et al., 2019] Lee, J., Xiao, L., Schoenholz, S., Bahri, Y., Novak, R., Sohl-Dickstein, J., and Pennington, J. (2019). Wide neural networks of any depth evolve as linear models under gradient descent. *Advances in neural information processing systems*, 32.

[Shallue et al., 2018] Shallue, C. J., Lee, J., Antognini, J., Sohl-Dickstein, J., Frostig, R., and Dahl, G. E. (2018). Measuring the effects of data parallelism on neural network training. *arXiv preprint arXiv:1811.03600*.

[Zhang et al., 2019] Zhang, G., Li, L., Nado, Z., Martens, J., Sachdeva, S., Dahl, G. E., Shallue, C. J., and Grosse, R. (2019). Which algorithmic choices matter at which batch sizes? insights from a noisy quadratic model. *arXiv preprint arXiv:1907.04164*.