Practical Methodology

Baraa Hassan and Keanu Buschbacher

February 9, 2021

UNIVERSITÄT BONN

Table of contents

Model Performance

Model Performance

Question

How to test your model performance?

Is it using:

1. **Cost Function**?
2. **Error Rate**?

Cost/Loss Function:

It gives you a numerical evaluation of how your model output is deviated from the target output, on the current state (train/dev./test).

Error Rate

It shows the rate of the dissimilarity between the model output and the target output.

The Cost Function's output is not an interpretable number for the accuracy, in the same time the Error Rate is not enough to address the performance of the model.

Let's see in the examples why.

Email Spam Detection System

Different kind of errors in performance

We can have 4 scenarios for this system output:

Spam as a Spam	VIP as a Spam
Spam as a VIP	VIP as an VIP

Email Spam Detection System has 2 kinds of mistakes:

1. Classifying legitimate message as a spam
2. Allow spam message to appear in the inbox

In case of performance evaluation the first mistake is catastrophic and needed much more to be prevented than the second one

Rare Disease Classifier

Imbalance data performance

Our classifier is a binary classifier for a rare disease that happens once in a million people.

Patient as a Patient	Non-patient as a Patient
Patient as a Non-patient	Non-patient as a Non-Patient

Our classifier can achieve accuracy of 99.9999%, by only output false for all input cases.

Although the classifier doesn't achieve the expected goal (detecting the patients catching this disease)

Confusion Matrix:

True Positive	False Positive
False Negative	True Negative

You should choose your performance metrics (that address your model accuracy) according to your expected goal from the model.

- To achieve our goal in Spam Detection System:
  - we can compute FPR (Type 1 Error) = $\frac{FP}{FP+TN}$ (to evaluate the first catastrophic mistake).
- To achieve our goal in Rare Disease Classifier:
  - we can compute Recall (True Positive Rate) = $\frac{TP}{TP+FN}$ (to get how many real patient were detected).
  - we can compute Precision (Positive Prediction Value) = $\frac{TP}{TP+FP}$ (to get how much of the detected values were right).

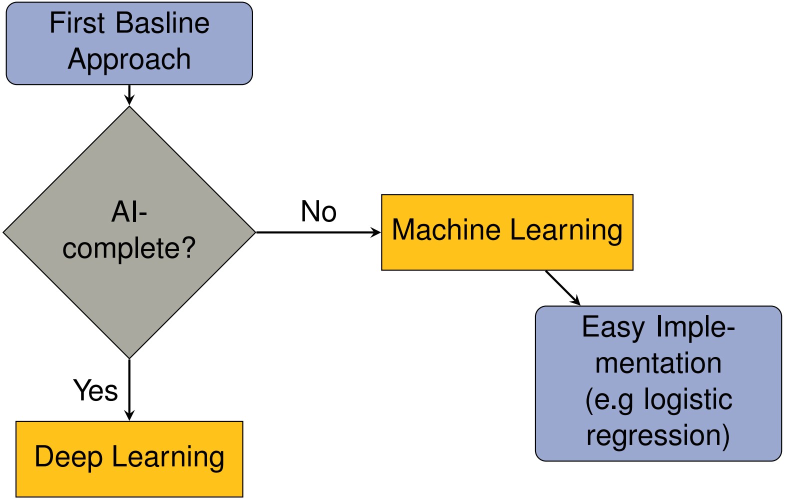
Precision and Recall are inversely proportional; in case of all the output are non-patient you will get a good precision but zero recall, and vice versa.

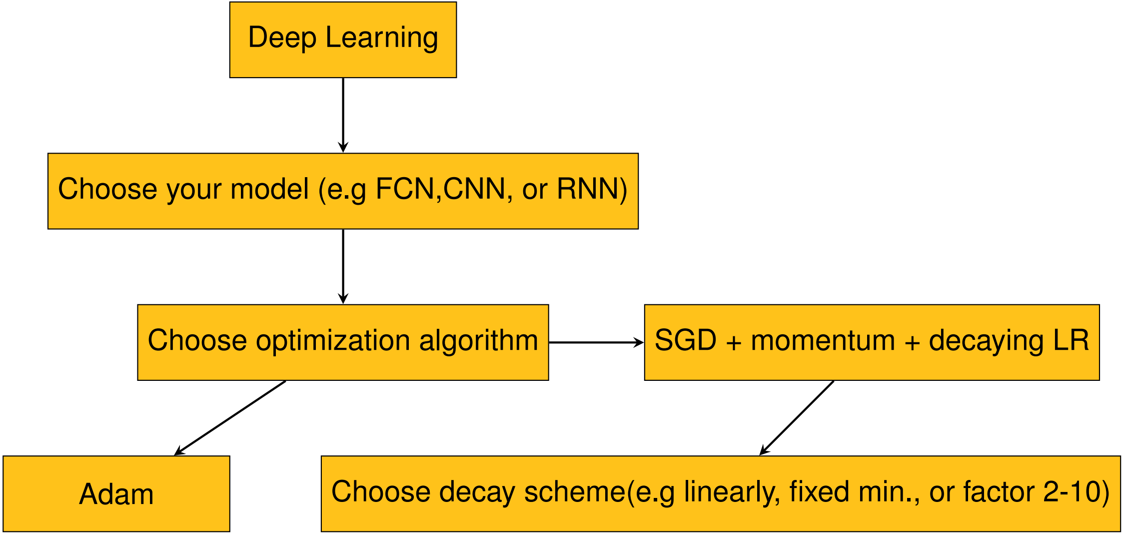
To get the mean of the two metrics you can compute F1 = $\frac{2*Precision*Recall}{Precision+Recall}$.

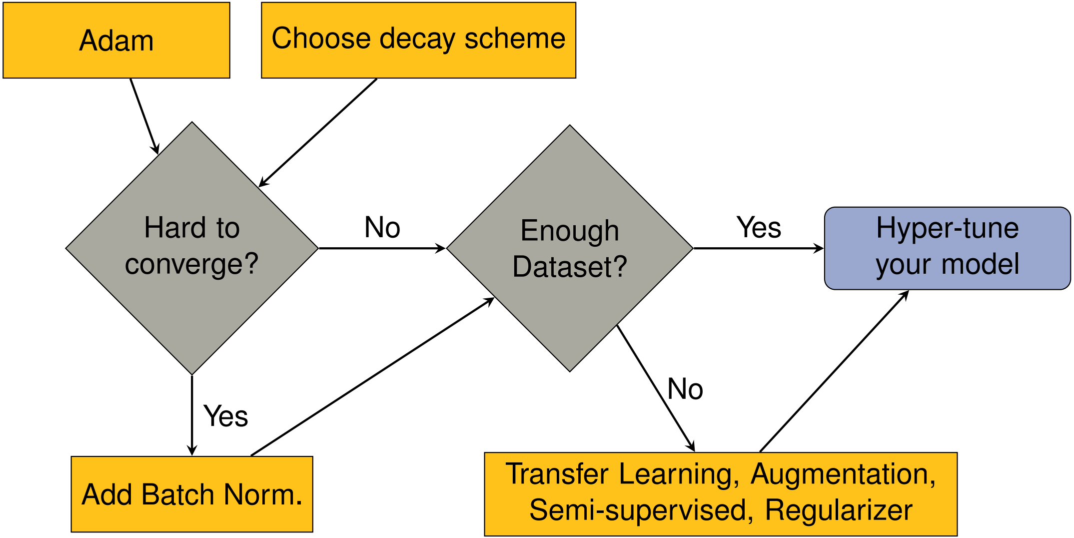
There are many different metrics that you can use to achieve your goal:

- For Classification:
  - FPR
  - FNR
  - Recall
  - Precision
- For Regression:
  - Mean Absolute Error(MAE)
  - Mean Squared Error(MSE)
  - Root Mean Squared Error(RMSE)

End-to-end System Development


First Basline Approach
AI-complete?
No
Machine Learning
Easy Imple-mentation (e.g logistic regression)
Yes
Deep Learning


Deep Learning
Choose your model (e.g FCN,CNN, or RNN)
Choose optimization algorithm
SGD + momentum + decaying LR
Adam
Choose decay scheme(e.g linearly, fixed min., or factor 2-10)



Adam
Choose decay scheme
Hard to converge?
No
Enough Dataset?
Yes
Hyper-tune your model
Yes
Add Batch Norm.
No
Transfer Learning, Augmentation, Semi-supervised, Regularizer

Determine Wether to Gather More Data

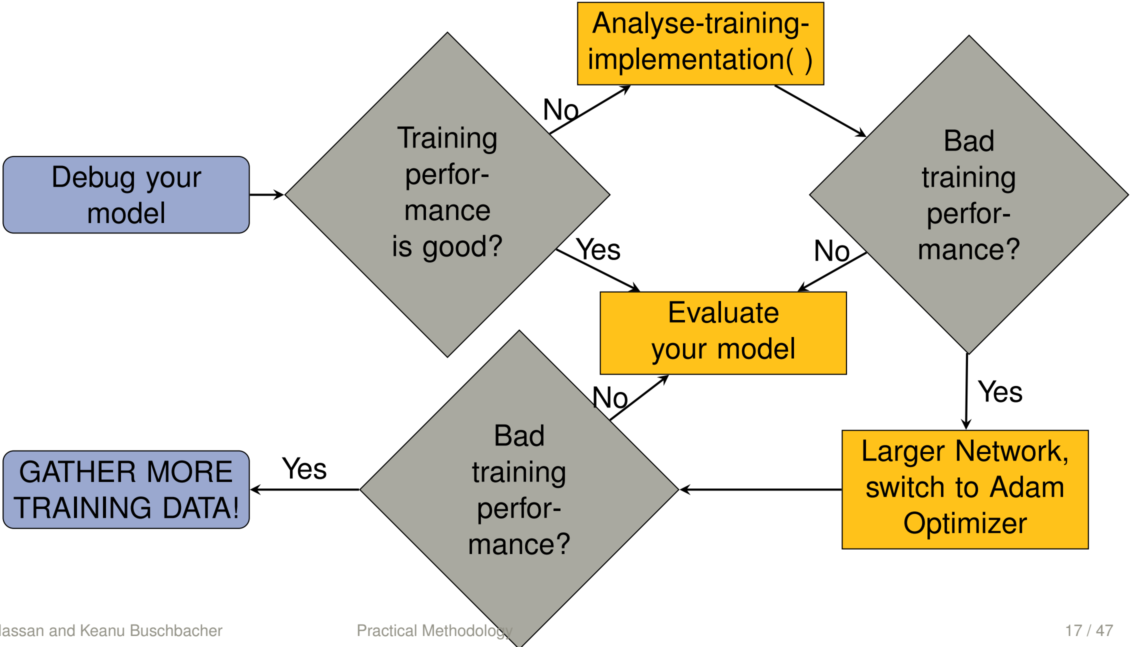
Whether to Gather More Data

Question

When to decide to collect more data ?

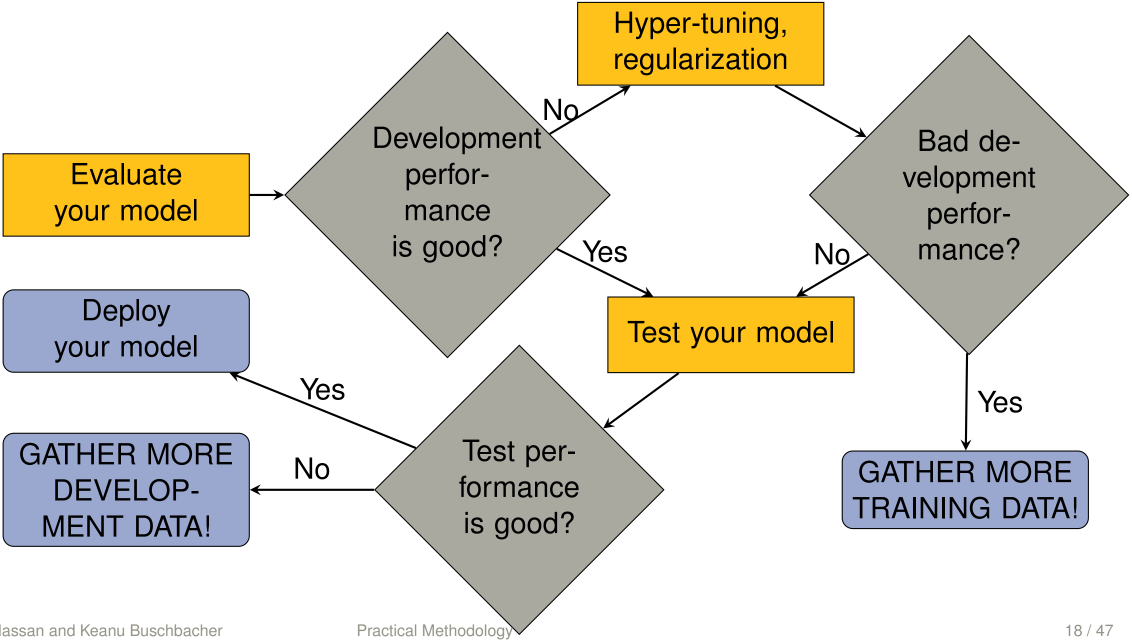
Taking into consideration the cost of collecting more data.

In the next flowchart we will discuss when to collect more data



Debug your model
Training perfor-mance is good?
No
Analyse-training-implementation( )
Bad training perfor-mance?
Yes
No
Evaluate your model
Yes
Larger Network, switch to Adam Optimizer
Bad training perfor-mance?
No
Yes
GATHER MORE TRAINING DATA!

Wether to Gather More Data



[NG, 2018]

Analysing Your Software Defects

To debug your train model method to check if you have software defect or under-fitting problem:

- Visualize the model in action
- Visualize the worst mistakes
- Fit a tiny dataset
- Compare back-propagated derivatives to numerical derivatives
- Mintor histograms of activations and gradient

Sometimes it is hard to find the source of the problem; as the machine learning models are composed of adaptive parts, if one went wrong the other parts can adapt and achieve roughly acceptable performance.

Selecting Hyperparameters

Selecting Hyperparameters

Hyperparameters control different aspects of how your model behaves.

1. **Costs**: time and memory requirements during training *(and inference)*
2. **Quality**: performance during training process and on new inputs

Some parameters actually influence both!

Strategies to tune Hyperparameters

Goal

Find hyperparameters that minimize the generalization error, s.t. they do not exceed our runtime and memory requirements.

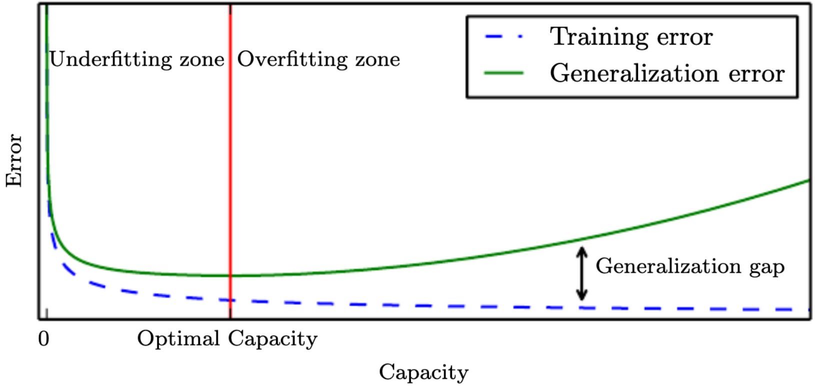
There are two kinds of strategies:

1. Manual hyperparameter tuning
   - No additional (explicit) computational complexity
   - Requires knowledge about hyperparameters...
2. Automatic hypterparameter tuning
   - Less need to understand hyperparameters
   - But computationally costly...

Manual Hyperparameter Tuning

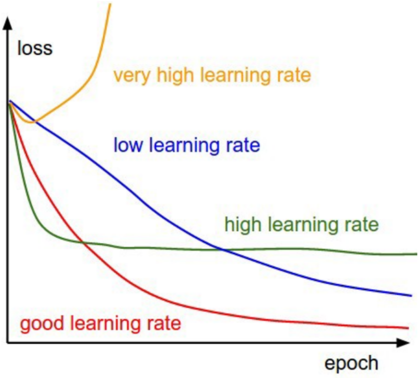
We want the capacity of our model to match the complexity of our task.
The effective model capacity consists of three factors:

Representational capacity	What functions can my model represent?
Cost function capacity	Can the learning algorithm minimize my cost function?
Regularization capacity	How strong do training and cost function regularize my model?



Training error
Generalization error
Underfitting zone
Overfitting zone
Error
Generalization gap
0
Optimal Capacity
Capacity

Tendency		Type		Tendency
underfitting / overfitting	← small value	Discrete / Cont. Hyperparameter	large value →	**overfitting / underfitting**
overfitting	← off	Binary Hyperparameter	on →	**underfitting**
		One-way Hyperparameter	→	**underfitting**



Source: CS231n Convolutional Neural Networks for Visual Recognition.[1]

The learning rate is a very important hyperparameter.

- too large: gradient descent can increase training error
- too low: slower training, more likely to get stuck in local minima

The model capacity is highest if the learning rate is set correctly!

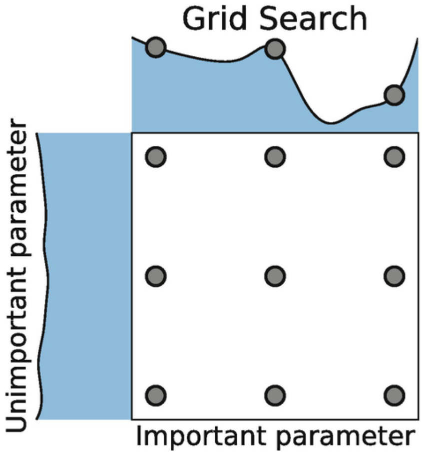
[1] `https://cs231n.github.io/assets/nn3/learningrates.jpeg`

Capacity		Hyperparameter		Capacity
		# Hidden Units	higher →	**increased**
decreased	← too low/high	Learning Rate	optimal →	**increased**
		Kernel size	higher →	**increased**
		Implicit zero padding	higher →	**increased**
decreased	← higher	Weight decay coefficient		
decreased	← higher	Dropout rate		

- Training error too large? **Increase model capacity!**
  - Are you using regularization? $\rightarrow$ Maybe use a bit less.
  - Is your optimization algorithm not performing correctly? $\rightarrow$ Fix it.
  - None of the above? $\rightarrow$ # Hidden Units Kernel size Padding
- Training error is fine, but test error is too large? We need to reduce the generalization gap. **Decrease model capacity!**
  - Add regularization capacity $\rightarrow$ Weight decay Dropout
  - Collect more training data

Automatic Hyperparameter Tuning

Choosing hyperparameters is also an optimization: find hyperparameter values that optimize an objective function, e.g. validation error.



Source: Bergestra et al. (2012).[2]

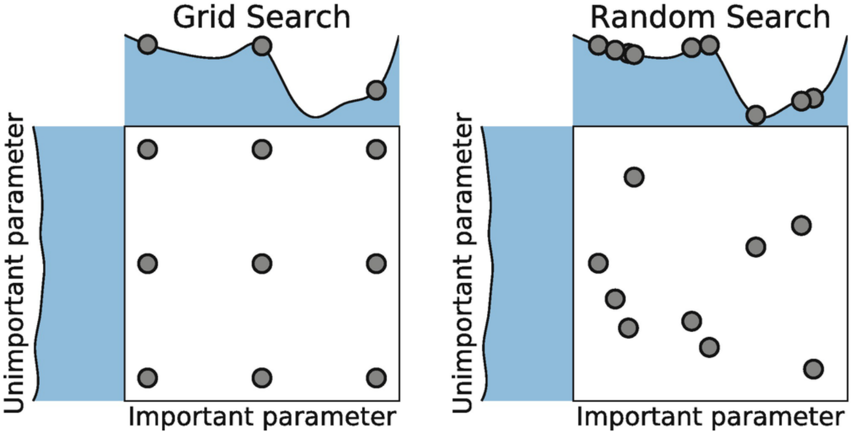
Grid Search

- For three or fewer hyperparameters
- Train on every combination of hyperparameter values
- Use best configuration according to validation error

[2] Bergstra, J., Bengio, Y.: Random search for hyper-parameter optimization. Journal of Machine Learning Research 13, 281–305 (2012)

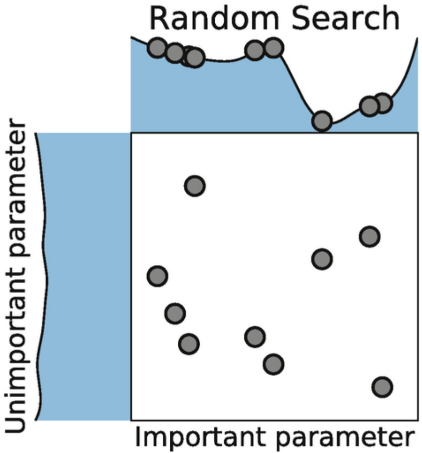
How to pick the hyperparameter values?

$\rightarrow$ Logarithmic scale: $\{0.1, 0.01, 10^{-3}, 10^{-4}, 10^{-5}\}$



Source: Bergestra et al. (2012).[3]

[3] Bergstra, J., Bengio, Y.: Random search for hyper-parameter optimization. Journal of Machine Learning Research 13, 281–305 (2012)



Source: Bergestra et al. (2012).[4]

Random Search

- Hyperparameter values sampled from random distribution
- Useful also for more than three hyperparameters
- Explores wider parameter space

[2] Bergstra, J., Bengio, Y.: Random search for hyper-parameter optimization. Journal of Machine Learning Research 13, 281–305 (2012)

Random search can be exponentially more efficient than grid search: it reduces validation error much faster w.r.t number of trials.

Trials are not ”wasted” if two hyperparameters give the same result.

Example: #Units $\in \{50, 100\}$ lead to same validation error:

#Units	LR	Kernel size	**Error**
50	0.01	5	**0.81**
100	0.01	5	**0.81**

Table: Grid search trials

#Units	LR	Kernel size	**Error**
50	0.009	3	**0.42**
100	0.012	5	**0.69**

Table: Random search trials

If gradient $\frac{\partial E}{\partial h}$ is available, we can just follow this gradient.

If not, we can train a model to optimize validation error with our hyperparameters as decision variables.

$\rightarrow$ Estimates validation error and uncertainty using Bayesian regression

Examples:

- Spearmint [Snoek et al., 2012]
- TPE [Bergstra et al., 2011]
- SMAC [Hutter et al., 2011]

Real-life Application of Practical Methodology

Goal

Assign digits to pictures of street numbers if model confidence $p(y|\ x) \geq t$ for some threshold t.

Figure: Street view transcription system. [Goodfellow et al., 2014]

Example: Multi-Digit Number Recognition

1. Choose performance metrics
    - Choose the metrics according to the project's business goal!
    - Here: maps require high, human-level accuracy
    - This meant a high threshold to accept results of the model

Metric is therefore: **coverage**, i.e. percentage of confidences above the threshold (goal: >95%)

Example: Multi-Digit Number Recognition

2. Establish baseline model
   - Try to iteratively improve the model!
   - Here: n different softmax units to predict n characters
   - each unit trained independently
   - $p(y|x)$ obtained by multiplying units together

Improvement idea: **use output layer/cost function that computes log-likelihood instead**

Example: Multi-Digit Number Recognition

Coverage was still below 90%. Is the problem under- or overfitting?

3. Debug model
   - Here: training and test error were identical
     $\rightarrow$ underfitting or problem with training data
   - Visualize model's worst mistakes

Example: Multi-Digit Number Recognition

Coverage was still below 90%. Is the problem under- or overfitting?

3. Debug model
   - Here: training and test error were identical
     → underfitting or problem with training data
   - Visualize model's worst mistakes:
     → Some images were cropped too tightly!

2 vs 239

Solution: **add margin of safety around crops** (+10% coverage)

Example: Multi-Digit Number Recognition

4. Adjust hyperparameters
   - Here: train and test error remained equal $\rightarrow$ underfitting
   - Model was made larger

Theme by @fseiffarth:

`https://github.com/fseiffarth/LatexBeamerThemeUniBonnStyle`

References

James Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. Algorithms for hyper-parameter optimization. In J. Shawe-Taylor, R. Zemel, P. Bartlett, F. Pereira, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 24, pages 2546–2554. Curran Associates, Inc., 2011. URL `https://proceedings.neurips.cc/paper/2011/file/86e8f7ab32cfd12577bc2619bc635690-Paper.pdf`.

Ian J. Goodfellow, Yaroslav Bulatov, Julian Ibarz, Sacha Arnoud, and Vinay Shet. Multi-digit number recognition from street view imagery using deep convolutional neural networks, 2014.

Frank Hutter, Holger H. Hoos, and Kevin Leyton-Brown. Sequential model-based optimization for general algorithm configuration. In Carlos A. Coello Coello, editor, *Learning and Intelligent Optimization*, pages 507–523, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg. ISBN 978-3-642-25566-3.

Andrew NG. Machine learning yearning. *URL¡ https://www.deeplearning.ai/programs/*, 1, 2018.

Jasper Snoek, Hugo Larochelle, and Ryan Adams. Practical bayesian optimization of machine learning algorithms. *Advances in Neural Information Processing Systems*, 4, 06 2012.

Thank You for Listening!