

# Training Neural Nets: a Hacker's Perspective

Sayak Paul | Deep Learning Associate at [PyImageSearch](https://pyimagesearch.com)

DevFest Warsaw&Radzymin 2019, December 07



2019



# Agenda

- The motivation
- Being thorough about training neural nets
  - Training a neural network
  - Maintaining a healthy prototyping process
  - Gradually increasing model complexity
  - ...
- Guiding lights

# The motivation

# The motivation



[Source](#)

# The motivation

***“Deep learning neural networks have become easy to define and fit, but are still hard to configure.”***

- Jason Brownlee, Machine Learning Mastery

# The motivation

## 2) Neural net training fails silently

When you break or misconfigure code you will often get some kind of an exception. You plugged in an integer where something expected a string. The function only expected 3 arguments. This import failed. That key does not exist. The number of elements in the two lists isn't equal. In addition, it's often possible to create unit tests for a certain functionality.

- Andrej Karpathy, Sr. Director of AI, Tesla

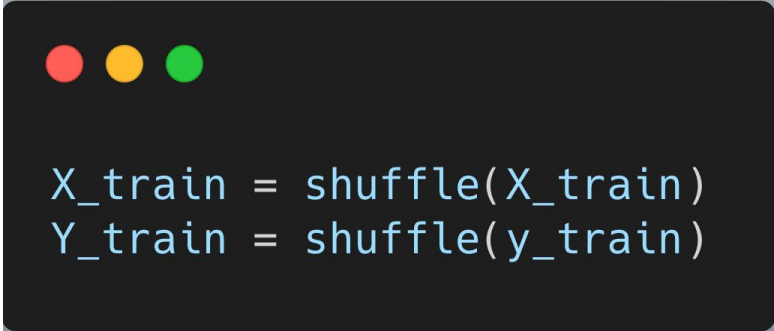
# Training a neural network

- Implementation bugs



# Training a neural network

- Implementation bugs




```
X_train = shuffle(X_train)  
Y_train = shuffle(y_train)
```

← Wrong shuffling!

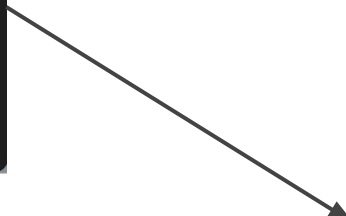


# Training a neural network

- Implementation bugs



```
X_train = shuffle(X_train)
Y_train = shuffle(y_train)
```



```
X_train, y_train = shuffle(X_train, y_train)
```



# Training a neural network

- Implementation bugs

Sex	Age	Has_Masters	Has_Bachelors	Bounties
0	23	1	1	23451
1	46	0	1	98731
0	21	0	1	12876
1	53	0	1	100234



```
scaler = StandardScaler()  
scaled_train = StandardScaler().fit_transform(data.values)
```

Feature  
interpretation

# Training a neural network

- Implementation bugs

Sex	Age	Has_Masters	Has_Bachelors	Bounties
0	23	1	1	23451
1	46	0	1	98731
0	21	0	1	12876
1	53	0	1	100234

```
scaled_train = StandardScaler().fit_transform(data[non_cat_feats].values)
```



# Training a neural network

- Implementation bugs
  - Wrong activation function & initialization

# Training a neural network

- Implementation bugs
  - Wrong activation function & initialization
    - **tanh + He**
    - **ReLU + Xavier**

# Training a neural network

- Implementation bugs
  - Wrong activation function & initialization
    - $\text{tanh} + \text{He} \rightarrow \text{tanh} + \text{Xavier}$
    - $\text{ReLU} + \text{Xavier} \rightarrow \text{ReLU} + \text{He}$

# Training a neural network

- Implementation bugs
  - Wrong activation function & initialization
  - Forgetting to *zero out gradients* (PyTorch)



```
for (...):  
    ...  
  
    log_ps = model(images)  
    loss = criterion(log_ps, labels)  
    loss.backward()  
    optimizer.step()  
  
    running_loss += loss.item()
```



No zero outs!

# Training a neural network

- Implementation bugs
  - Wrong activation function & initialization
  - Forgetting to *zero out gradients* (PyTorch)

```
for (...):  
    ...  
  
    optimizer.zero_grad()  
  
    log_ps = model(images)  
    loss = criterion(log_ps, labels)  
    loss.backward()  
    optimizer.step()  
  
    running_loss += loss.item()
```





# Training a neural network

- Implementation bugs
- Model's sensitivity towards hyperparameter choices

# Training a neural network

- Implementation bugs
- Model's sensitivity towards hyperparameter choices
  - Too high learning rate leads to numerical instability
    - Loss quantity coming out as **NaNs**

# Training a neural network

- Implementation bugs
- Model's sensitivity towards hyperparameter choices
  - Too high learning rate leads to numerical instability
  - Too few epochs
    - Network not trained enough

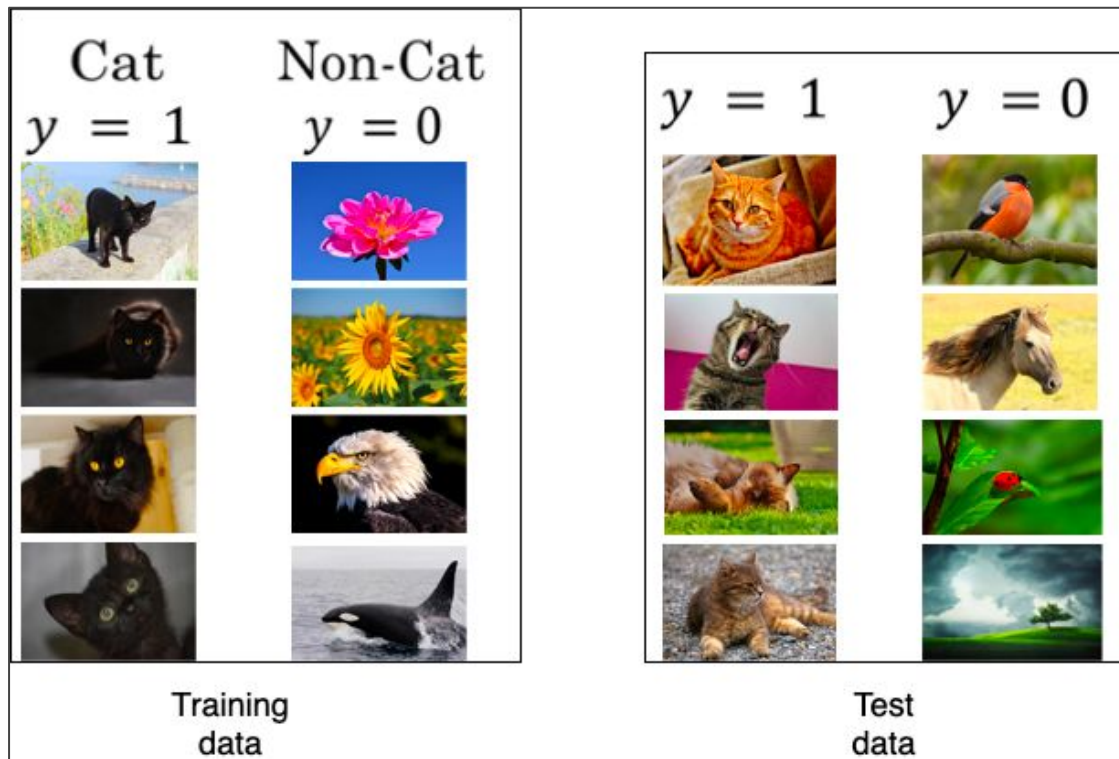
# Training a neural network

- Implementation bugs
- Model's sensitivity towards hyperparameter choices
  - Too high learning rate leads to numerical instability
  - Too few epochs
  - Too large batch size for a small dataset

# Training a neural network

- Implementation bugs
- Model's sensitivity towards hyperparameter choices
- Dataset construction and others
  - Dataset distribution

# Training a neural network



[Source](#)

[sayak.dev](http://sayak.dev)

# ... and just for fun



A sample image from the training set

# ... and just for fun



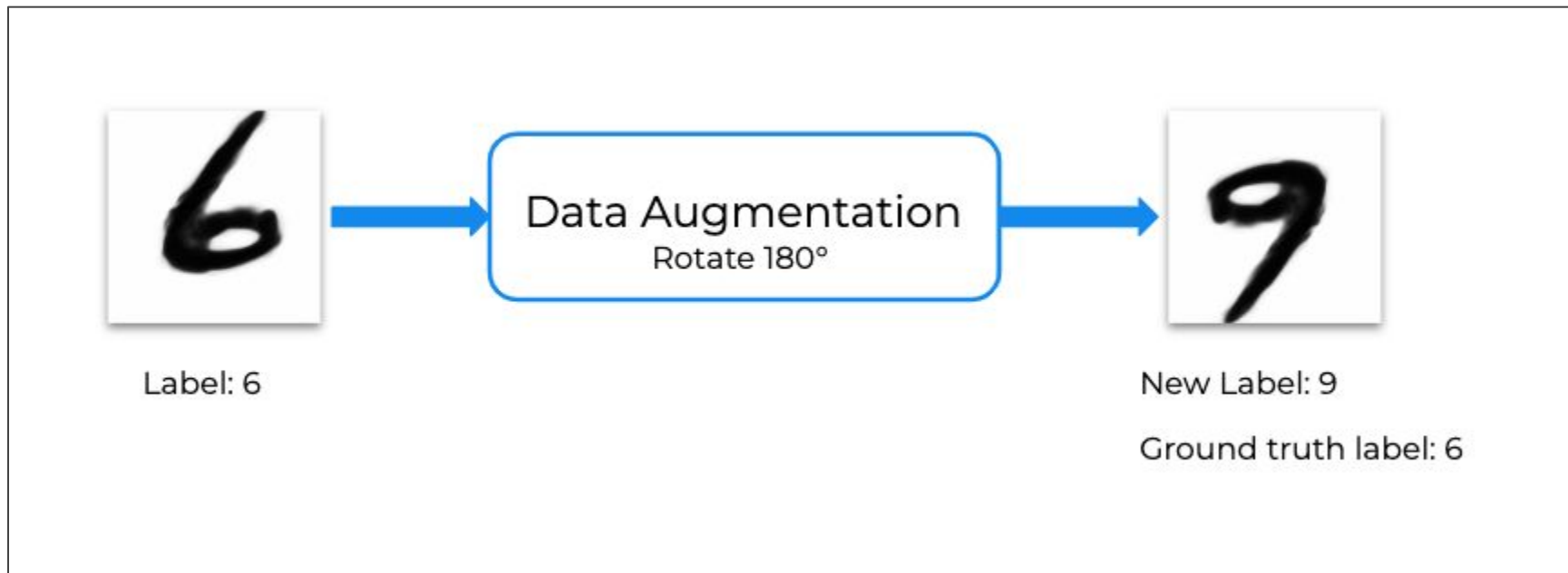
A corner case



# Training a neural network

- Implementation bugs
- Model's sensitivity towards hyperparameter choices
- Dataset construction and others
  - Dataset distribution
  - Effects of random data augmentation

# Training a neural network



[Source](#)

# Training a neural network

- Implementation bugs
- Model's sensitivity towards hyperparameter choices
- Dataset construction and others
  - Dataset distribution
  - Effects of random data augmentation
  - Wrong normalization statistics
  - Label noise
  - Class imbalance

# Maintaining a healthy prototyping process

- Write code quickly
  - Reuse existing codebases for quick baselines
  - But be very careful

# Maintaining a healthy prototyping process

- Write code quickly
  - Reuse existing codebases for quick baselines
  - But be very careful

## **Writing code quickly - Use a framework!**

- Don't start from scratch! Use someone else's components.

# Maintaining a healthy prototyping process

- Write code quickly
- Run experiments and keep track of what you tried

# Maintaining a healthy prototyping process

- Write code quickly
- Run experiments and keep track of what you tried

Experimenter	git SHA	Background Search Method	Model	Dataset	Train Acc	Validation Acc	Notes
Pradeep	fc8d6ca3	Lucene	QAMNS (50d)	Intermediate	0.3114	0.3045	patience=20
Pradeep	fc8d6ca3	Lucene	QAMNS (300d)	Intermediate	0.8317	0.3864	patience=20
Pradeep	fc8d6ca3	BOW-LSH question+answers Glove 50d	QAMNS (50d)	Intermediate	0.3008	0.35	patience=20
Pradeep	fc8d6ca3	BOW-LSH question+answers Glove 50d	QAMNS (300d)	Intermediate	0.7466	0.4227	patience=20
Pradeep	fc8d6ca3	BOW-LSH question+answers Glove 300d	QAMNS (50d)	Intermediate	0.3946	0.3591	patience=20
Pradeep	fc8d6ca3	BOW-LSH question+answers Glove 300d	QAMNS (300d)	Intermediate	0.7311	0.4227	patience=20
Pradeep	fc8d6ca3	BOW-LSH+IDF question+answers Glove 300d	QAMNS (300d)	Intermediate	0.7446	0.4227	patience=20
Pradeep	fc8d6ca3	BOW-LSH+IDF question+answers Paragram 300d	QAMNS (300d)	Intermediate	0.7853	0.3955	patience=20
Pradeep	fc8d6ca3	Lucene	QAMNS (300d)	SciQ	0.5551	0.571	patience=6
Pradeep	fc8d6ca3	BOW-LSH question+answers Glove 300d	QAMNS (300d)	SciQ	0.5434	0.524	patience=6

[Source](#)

# Maintaining a healthy prototyping process

- Write code quickly
- Run experiments and keep track of what you tried
- Analyze model behavior. Did it do what you wanted?



# Overfitting a single batch of data

- The loss could go up instead of down

# Overfitting a single batch of data

- The loss could go up instead of down
- The loss could go down for a while, then explode

# Overfitting a single batch of data

- The loss could go up instead of down
- The loss could go down for a while, then explode
- The loss could oscillate across a region

# Overfitting a single batch of data

- The loss could go up instead of down
- The loss could go down for a while, then explode
- The loss could oscillate across a region
- The loss could get down to a fixed quantity (0.01, for example) and not get any better than that

# Model complexity as a $f(\text{order})$

Deciding on a model architecture:

# Model complexity as a $f(\text{order})$

Deciding on a model architecture:

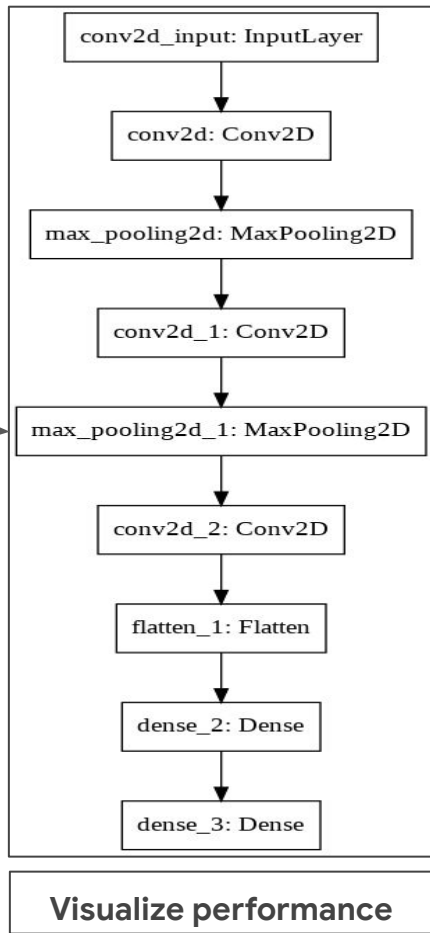
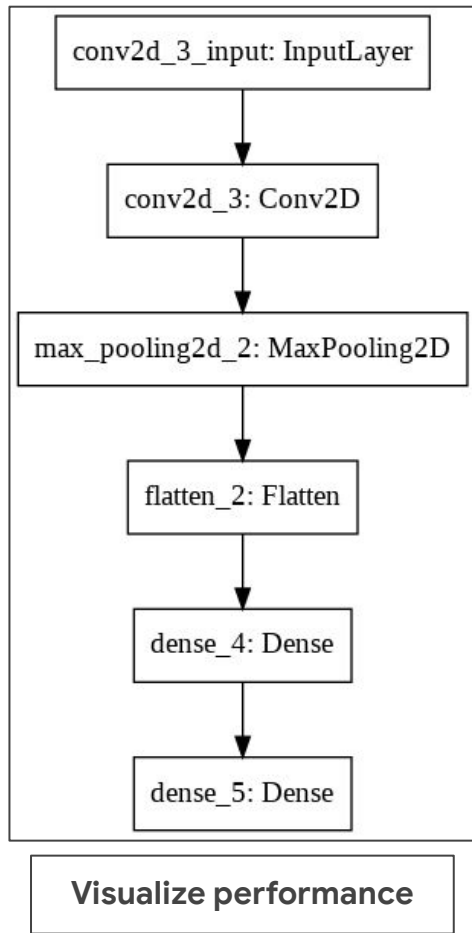
- Time to train the network
- Size of the final network
- Inference speed
- Accuracy

# Model complexity as a $f(\text{order})$

Some points to remember here:

- Ramping up the model complexity gradually

# Model complexity as a $f(\text{order})$



Deep learning is a shout in the void and the oblivion is inevitable!



# Model complexity as a $f(\text{order})$

Some points to remember here:

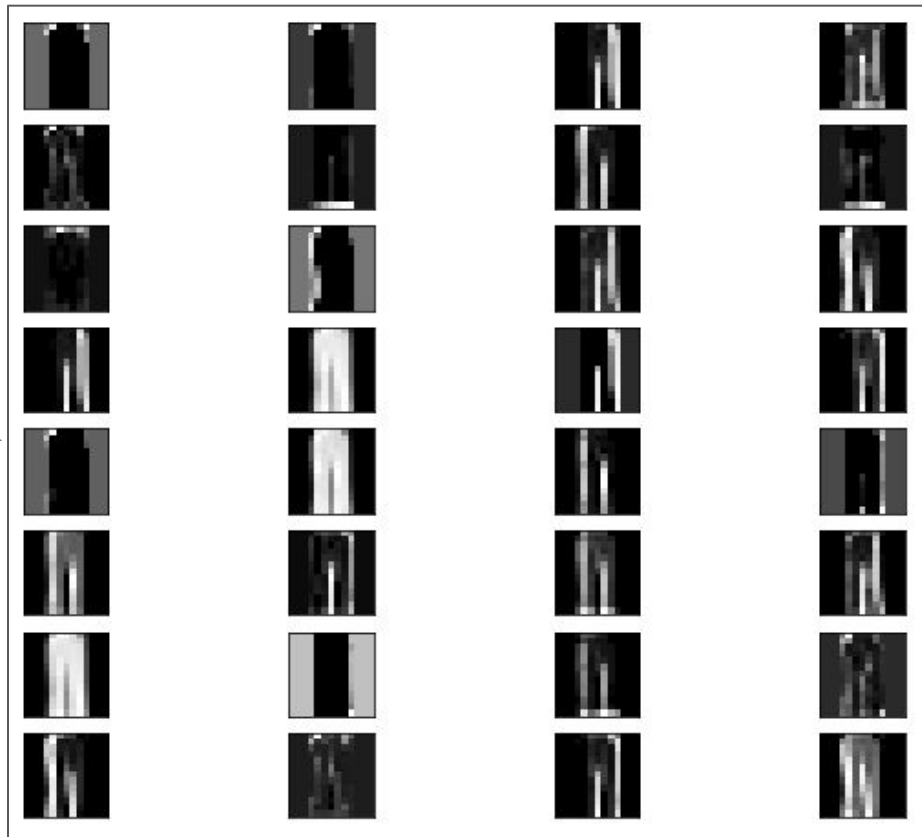
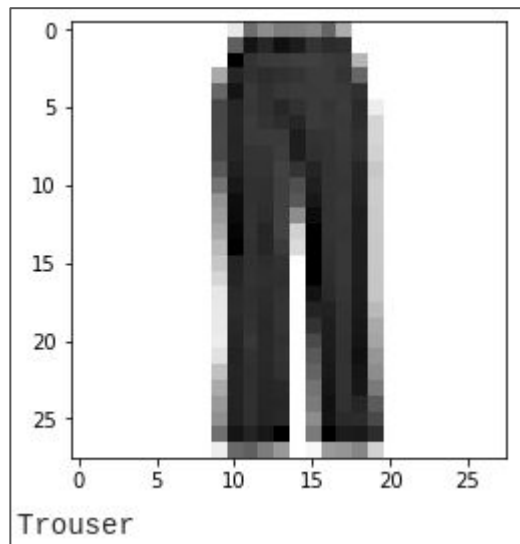
- Ramping up the model complexity gradually

# Model complexity as a $f(\text{order})$

Some points to remember here:

- Ramping up the model complexity gradually
- Experimentation with several random subsets
- Human evaluation
- Visualizing the intermediate activations of the model

# Model complexity as a $f(\text{order})$



# Chasing the hyperparameters

- Declarative configuration

# Chasing the hyperparameters

- Declarative configuration

```
floyd run trainer.py \  
  --gpu2 \  
  --env tensorflow-1.14 \  
  --data sayak/datasets/imdb:imdb \  
  'python trainer.py \  
    --model=bert \  
    --model_type=bert-base-uncased \  
    --problem=sentiment \  
    --data_dir=/floyd/input/imdb \  
    --train \  
    --eval \  
    --max_seq_length=128 \  
    --batch_size=256 \  
    --learning_rate 2e-5 \  
    --num_train_epochs= 10 \  
    --output_dir ./output-bert-uncased  
  ...'
```



```
floyd run --task train
```

```
# floyd.yml  
env: tensorflow-1.14  
  
task:  
  train:  
    machine: gpu2  
    description: sentiment with bert-un  
    input:  
      - source: sayak/datasets/imdb  
        destination: imdb  
    command: trainer.py \  
      --model=bert \  
      --model_type=bert-base-uncased \  
      ...
```

# Chasing the hyperparameters

- Declarative configuration
  - Use of **Hyperparameter Sweeps** with [Weights & Biases](#)
















# Chasing the hyperparameters

- Declarative configuration
  - Use of **Hyperparameter Sweeps** with [Weights & Biases](#)

```
program: train.py
method: bayes
metric:
  name: val_loss
  goal: minimize
parameters:
  learning-rate:
    min: 0.001
    max: 0.1
  optimizer:
    values: ["adam", "sgd"]
```

# Chasing the hyperparameters

- Declarative configuration
  - Use of **Hyperparameter Sweeps** with [Weights & Biases \(wandb\)](#)

<input type="checkbox"/>  Name (...)	State	Notes	User	Tags	Created	Runtime	epochs	layers	accuracy	epoch	examples	graph	loss
  0...	crashed	<a href="#">Add notes</a>	saya...	—	4d ago	23s	5	32	0.8683	3	Image	graph	0.3633
  a...	finished	<a href="#">Add notes</a>	saya...	—	4d ago	35s	5	32	0.8795	4	Image	graph	0.3328
  d...	finished	<a href="#">Add notes</a>	saya...	—	5d ago	9m 36s	5	128	0.8918	4	Image	graph	0.2932
  1...	finished	<a href="#">Add notes</a>	saya...	—	4d ago	42s	5	96	0.8895	4	Image	graph	0.3023
  z...	finished	<a href="#">Add notes</a>	saya...	—	4d ago	34s	5	128	0.8906	4	Image	graph	0.2958
  1...	finished	<a href="#">Add notes</a>	saya...	—	4d ago	39s	5	64	0.8869	4	Image	graph	0.3093
  p...	finished	<a href="#">Add notes</a>	saya...	—	4d ago	42s	5	256	0.895	4	Image	graph	0.2834

Hyperparameter sweep results



# Chasing the hyperparameters

- Declarative configuration
  - Use of **Hyperparameter Sweeps** with [Weights & Biases](#) (**wandb**)
    - The experiments are hosted by **wandb** as a cloud service
    - Can also be run locally via Python API

# Chasing the hyperparameters

- Declarative configuration
- Organizing the hyperparameter tuning process

# Chasing the hyperparameters

- Declarative configuration
- Organizing the hyperparameters' search process

Hyperparameter	Approximate sensitivity
Learning rate	High
Optimizer choice	Low
Other optimizer params (e.g., Adam beta1)	Low
Batch size	Low
Weight initialization	Medium
Loss function	High
Model depth	Medium
Layer size	High
Layer params (e.g., kernel size)	Medium
Weight of regularization	Medium
Nonlinearity	Low

[Source](#)

# Going beyond ...

- Model ensembling

# Going beyond ...

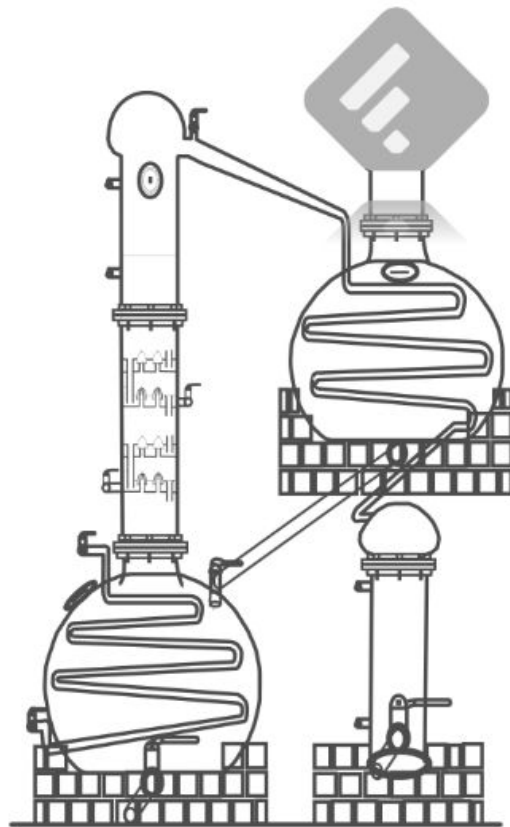
- Model ensembling
- Knowledge distillation

# Going beyond ...

- Model ensembling
- Knowledge distillation
- Lottery ticket hypothesis

# Going beyond ...

- Model ensembling
- Knowledge distillation
- Lottery ticket hypothesis
- Model quantization



# Summary

- Training (and debugging) neural nets
- Healthy model prototyping
- The idea of overfitting a single batch of data
- Practical hyperparameter tuning
- Squeezing the best out of a network



# References

- [Training Neural Nets: a Hacker's Perspective](#)
- [A Recipe for Training Neural Networks](#)
- [Troubleshooting Deep Neural Networks](#)
- [Deep Learning from the Foundations](#)

# References

- [Training Neural Nets: a Hacker's Perspective](#)
- [A Recipe for Training Neural Networks](#)
- [Troubleshooting Deep Neural Networks](#)
- [Deep Learning from the Foundations](#)

Slides are available here: <http://bit.ly/dfw-sayak>

# See you next time



Find me here:  
[@RisingSayak](https://twitter.com/RisingSayak)

Thank you very much :)



Experts

