Welcome to Week 11 Lecture 2!

Data Science in Python & Machine Learning

Neural Networks Review Poll:

1. Forward Propagation...
2. Backward Propagation...

Learning Objectives

After this lesson you will be able to:

- ❏ Identify bias and variance in neural networks
- ❏ Strategically tune a neural network.

Building & Tuning Neural Networks

Cheatsheet: Neural Networks in Keras

(Copy or screenshot this slide for reference)

A Keras model follows these main steps:

1. Instantiate the model type (Sequential for our models)
2. Add each layer in the order you want them to run
    a. Each layer needs a **number of nodes** and an **activation function (hyperparameters)**
    b. 1st layer must have defined input size (input_dim=X_train.shape[1])
    c. Last layer:
        i. Nodes = 1 for regression or binary classification, OR the number of target classes for multiclass classification.
        ii. Activation must be appropriate to the problem.
            1. 'linear' for regression
            2. 'sigmoid' for binary classification
            3. 'softmax' for multiclass classification
3. Compile model
    a. Loss appropriate for problem (MSE, BCE, or 'categorical_crossentropy')
    b. (optional but recommended) Optimizer ('adam' is good!)
    c. (optional but recommended) additional metrics appropriate for the problem type. You can find list of metrics here
4. Fit model
    a. Training set
    b. Validation set
    c. Epochs
    d. (optional) Callbacks

Tuning Neural Networks

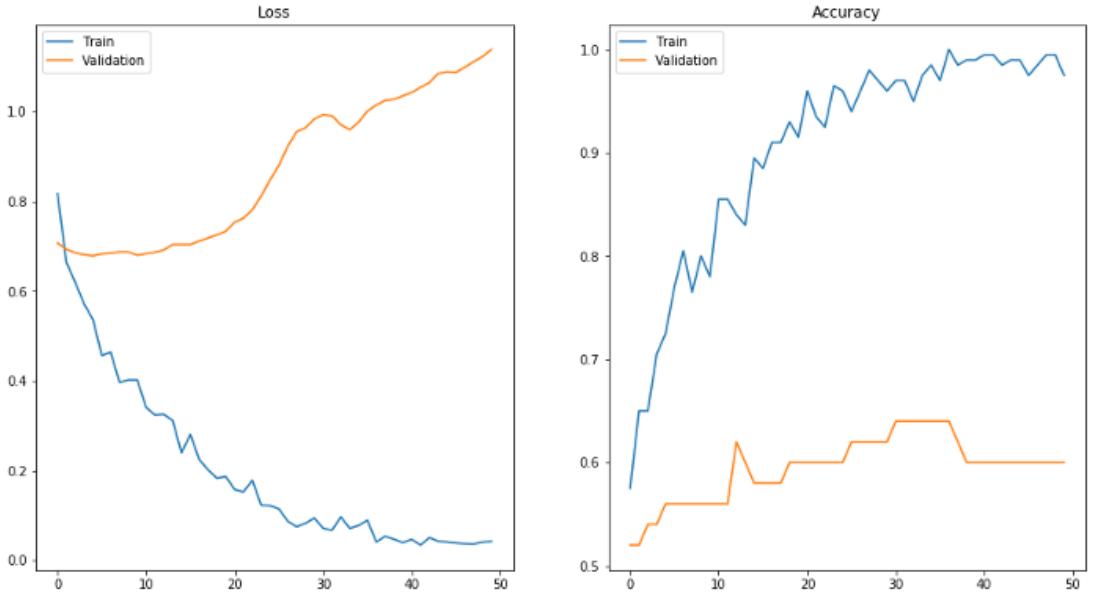
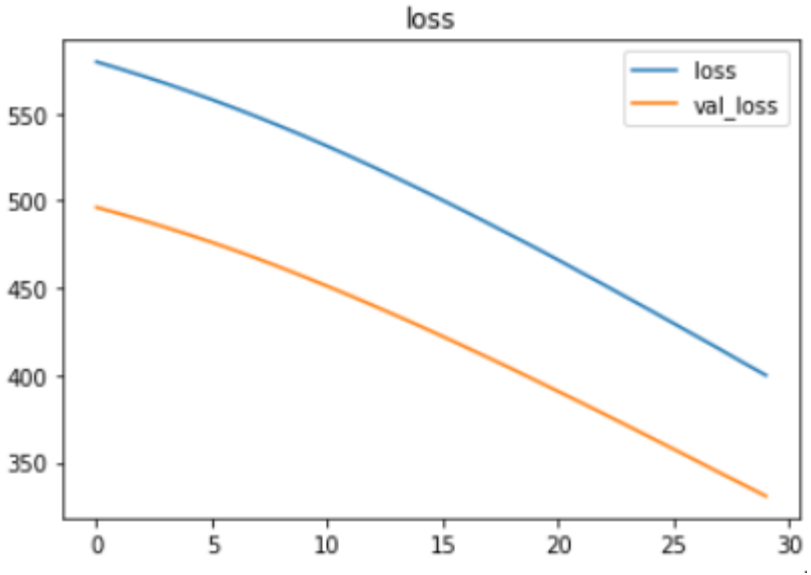


Image Source

Underfitting (High Bias)

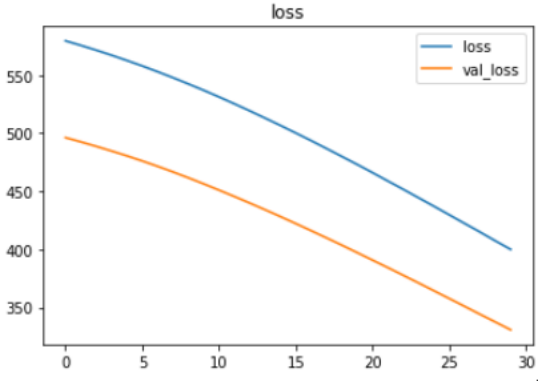
A model that is **too simple** or **trained for too few epochs** won't learn to predict a dataset well.



Attacking High Bias

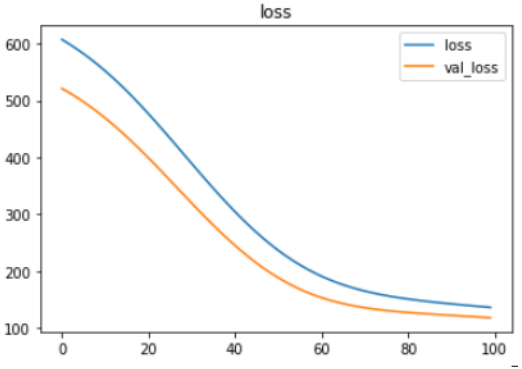
If model still seems to be learning:

- Increase epochs
- Add nodes
- Add Layers



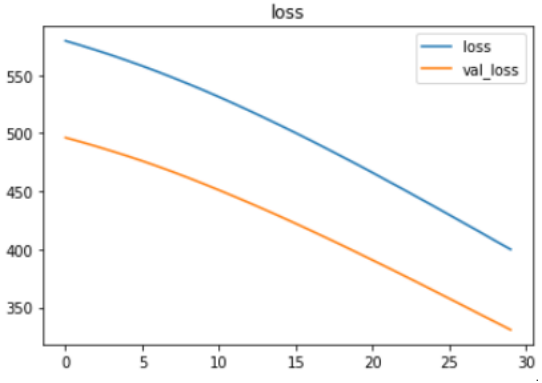
If learning has leveled off:
increase model complexity

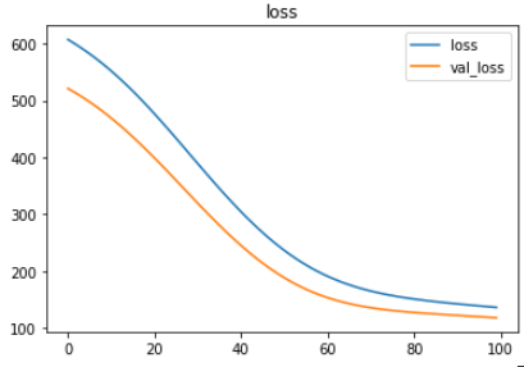
- Add nodes
- Add layers



Attacking High Bias

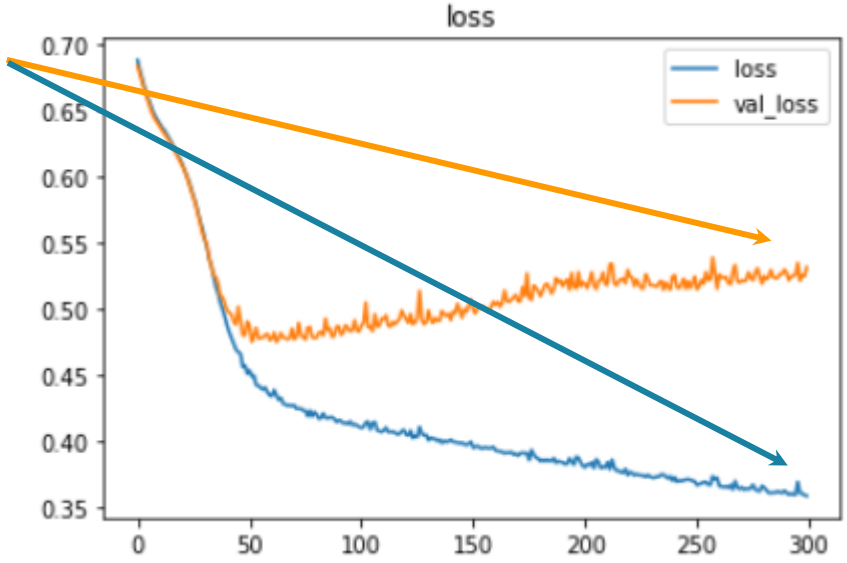
If Variance is low, attack the bias!





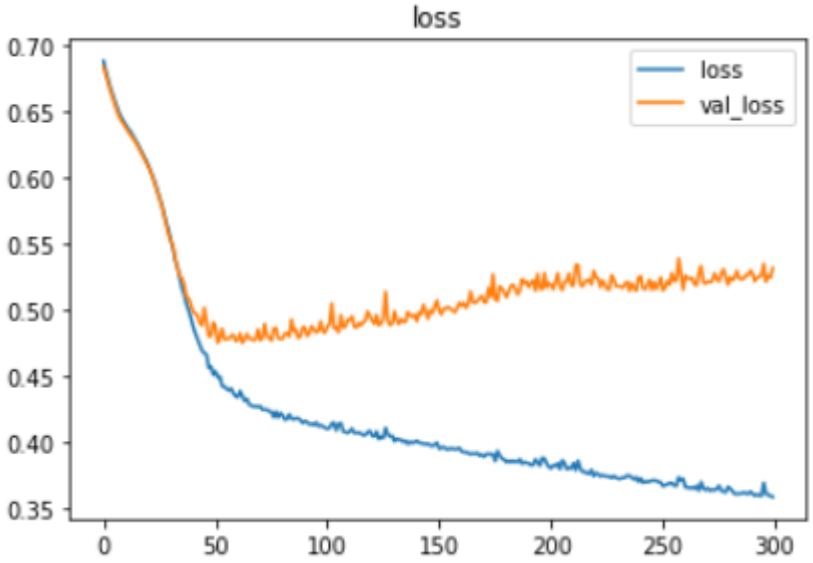
Overfitting (High Variance)

- High Variance: model performs worse on test data than on training data.



Overfitting (High Variance)

- Neural networks are prone to overfitting
- **Attacking High Variance:**
  - Dropout Layers
  - Early Stopping
  - L1 and L2 Regularization
  - Decrease model complexity



Dropout Layers

What it does:

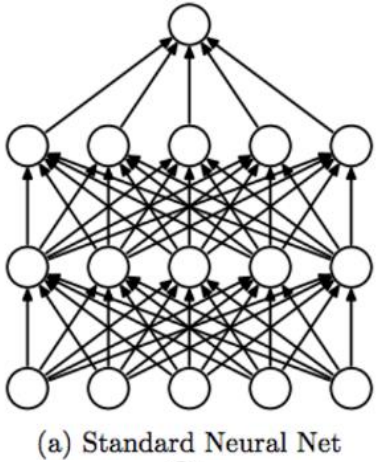
A dropout layer ‘turns off’ nodes at random in the layer before fit.

It turns off **different** nodes on each epoch.

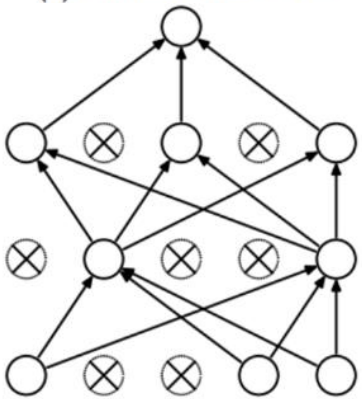
Why it works:

A network can become overly reliant on a few particular nodes which pick up very specific patterns.

Dropout forces a network to distribute the learning across many nodes and find more generalizable patterns



(a) Standard Neural Net



(b) After applying dropout.

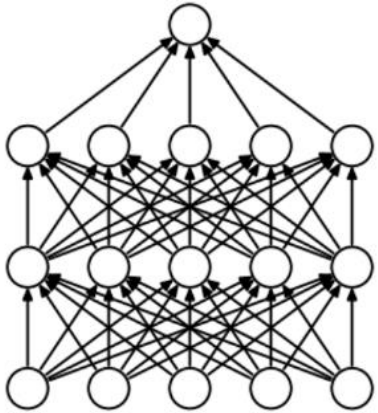
Image Source:
Srivastava, Nitish, et al. "Dropout: a simple way to prevent neural networks from overfitting", JMLR 2014

Dropout Layers

How to do it:

```
1   # create and compile the model
2   from tensorflow.keras.layers import Dropout
3
4   model = keras.Sequential()
5   model.add(Dense(10, input_dim=X_train_processed.shape[1],
6             activation = 'relu'))
7   model.add(Dropout(.2))
8   model.add(Dense(5, activation = 'relu'))
9   model.add(Dense(1, activation = 'sigmoid'))
10
11  model.compile(optimizer = 'adam', loss = 'bce', metrics = 'accuracy')
```

Percentage of nodes to ‘drop’ on each epoch



(a) Standard Neural Net

(b) After applying dropout.

Callbacks: Early Stopping

What it does:
Callbacks are functions that can be called during training. They checks something about the training between epochs.

Early stopping stops training early.

Why it works:
One source of overfitting is **over-training.** After a certain number of epochs a model will often begin to fit too tightly to training data. It learns TOO well. Early stopping stops the model before that can happen.

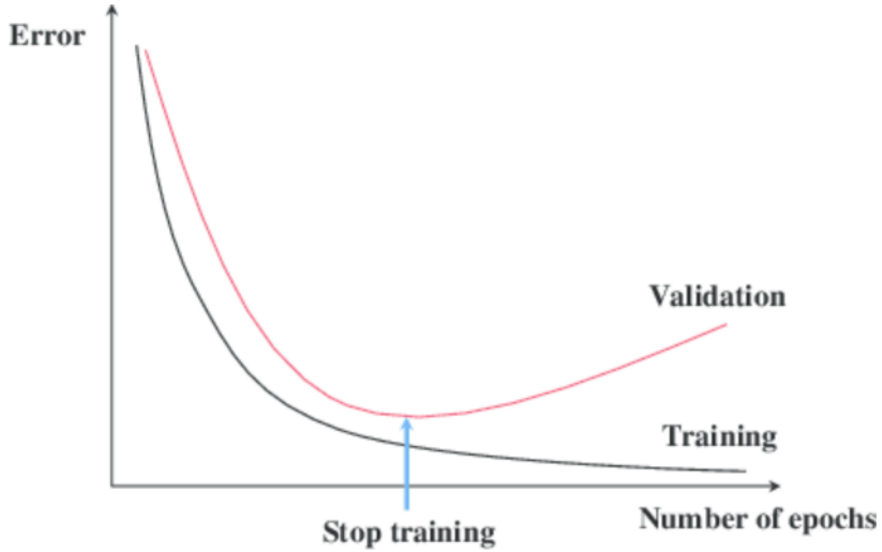


Image Source

Callbacks: Early Stopping

```
from tensorflow.keras.callbacks import EarlyStopping
early_stopping = EarlyStopping(patience=3)
# train the model and save the history
history = model.fit(X_train_processed, y_train,
          validation_data=(X_test_processed, y_test),
          epochs=10,
          callbacks=[early_stopping])
```

List of callbacks in fit call (There are many other callbacks you can use!)

If no reduction in validation loss in 3 epochs, stop training

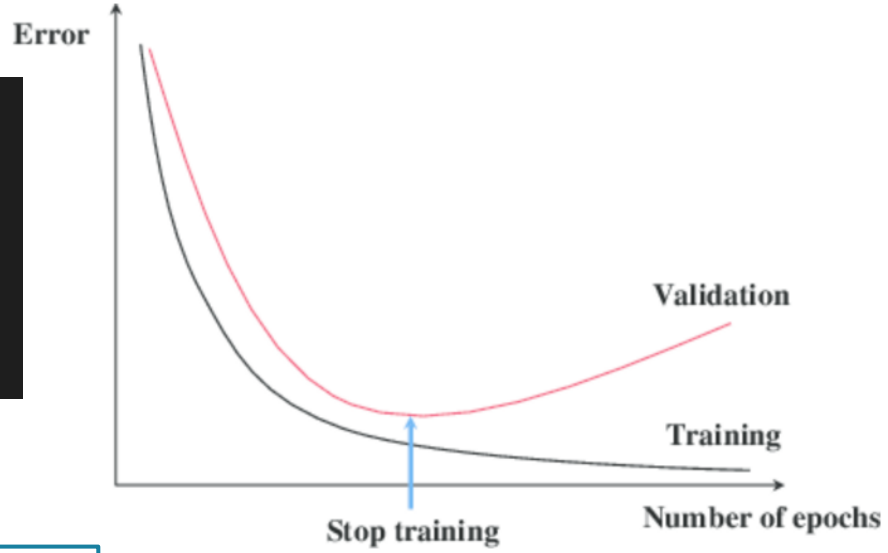


Image Source

Advanced Technique: L1 and L2 Regularization

What it does

- Neural network node = linear regression model
- L1 and L2 regularization to limit the change in weight values by applying a ‘penalty’ term.

Why it works

- Weights can get too large
  - Model puts too much emphasis on a small subset of features
- Regularization L1 and L2 prevent this.

Advanced Technique: L1 and L2 Regularization

How to do it:

```
from tensorflow.keras import layers
from tensorflow.keras import regularizers

layer = layers.Dense(
    units=64,
    kernel_regularizer=regularizers.l1_l2(l1=1e-5, l2=1e-4),
    bias_regularizer=regularizers.l2(1e-4),
    activity_regularizer=regularizers.l2(1e-5)
)
```

Image Source: Keras Documentation

Reduce Model Complexity

What it does:
Reduce the number of nodes and/or layers in your model.

Why it works:
One source of overfitting is a model that is too complex. An overly complex model will fit to the noise in the data rather than the true signal.

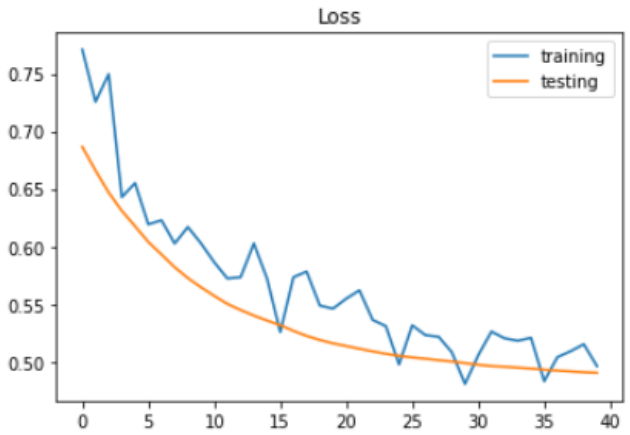
How it works:
Create a model with fewer nodes and/or layers.
(Remember to just change one thing at a time when experimenting)

Remember: Always evaluate each model with MULTIPLE metrics.

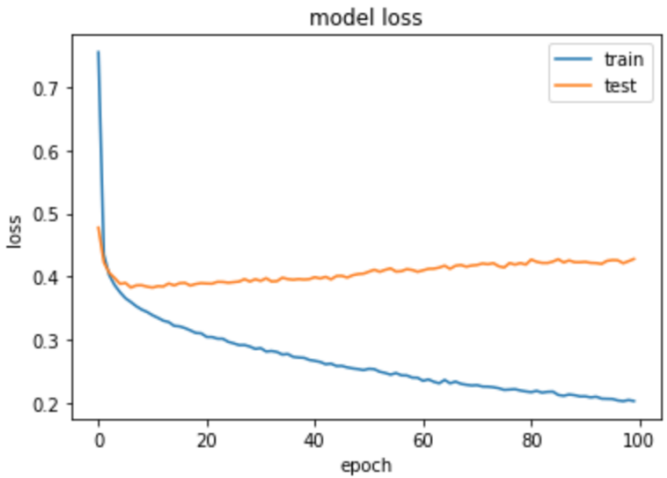
The best model is the model with the best metrics on test data,

regardless of overfitting!

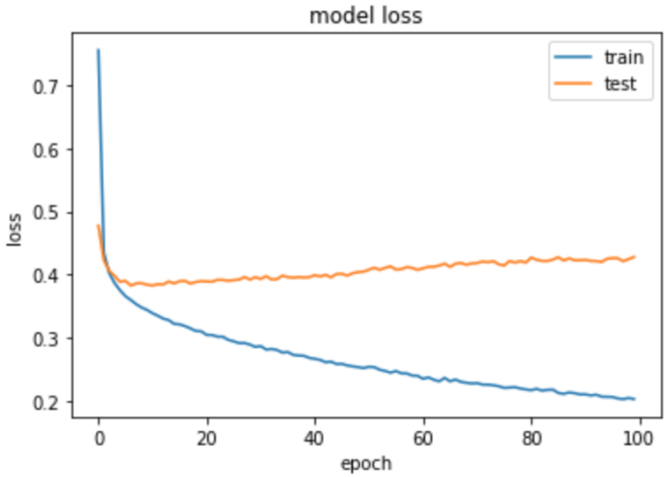
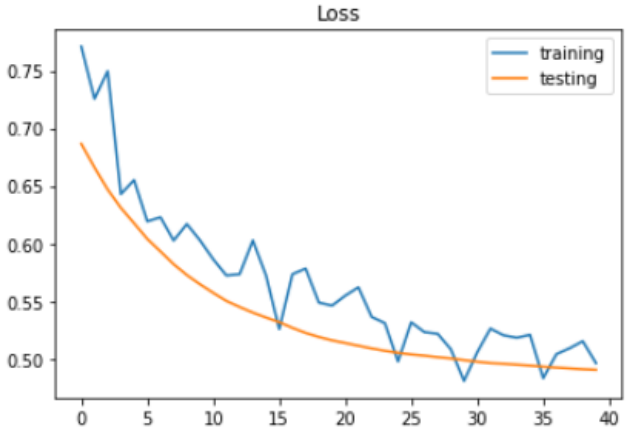
Attack Bias or Attack Variance? (POLL)

Attack Bias or Variance?	How?
bias	Increase complexity (nodes)
	Increase complexity (layers)
	Train longer

Attack Bias or Attack Variance? (POLL)

Attack Bias or Variance?	How?
Variance	Reduce complexity (nodes and layers)
	dropout
	L1, l2 Early stopping

Attack Bias or Attack Variance?

High Variance	Low Variance
Attack the Variance	Attack the Bias
	

Tuning a Model:

Strategies:

1. More or fewer nodes
2. More or fewer layers
3. More or fewer epochs
4. Regularization
    a. L1 or L2
    b. Dropout layers
    c. Early Stopping
5. Change activation functions
6. Change optimizers
7. And much more!

Let’s Tune a Model Together! (Code-along)

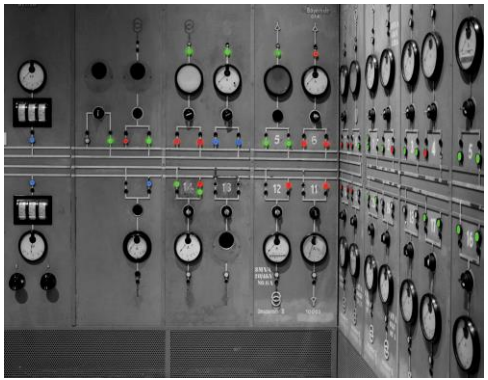
Code-along Notebook

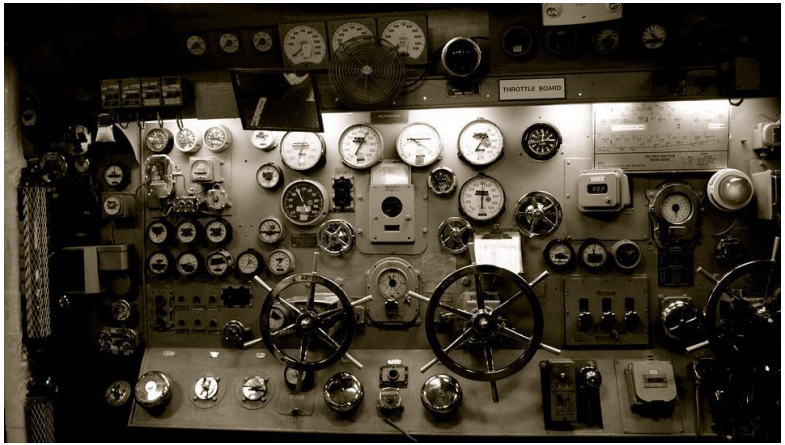
Your Challenge: Tune a Neural Network!

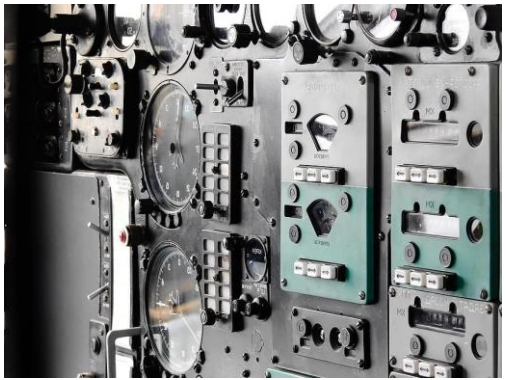
Your task will be to tune the classification model we created in the last lecture.

1. Make a copy of the notebook
2. Run the notebooks
3. Examine the learning plot and decide what you will try with the next model version.
4. (optional) copy/paste the model code from above and make changes to improve test accuracy

Here is the starter notebook







Announcements: Belt Exam

- Belt exam code will be given to the eligible students before Friday evening
- To receive a code for this weekend: you must submit by Friday 9 am Pacific:
  - Week 9 (including all resubmits) and Week 10 assignments
- You will be given belt color based on
  - Black Belt: Your Score >= 9.5
  - Red Belt: 8.0 <= Your Score <= 9.5
  - Retake/Rollback: Your Score < 8.0

Everything covered in weeks 9 through 11 will be included in the belt exam, so you need to review it all.

Announcements: Presentations

- Presentations on Thursday next week.
- You will find out the order in class on Thursday, so everyone must be ready!
- You will have a max of 5 minutes (we will have a timer) so please practice!
- Should be aimed at a Non-Technical Audience!
  - What would a:
    - non-technical,
    - non-data scientist,
    - non-statistician
    - NOT understand?

TAs can give feedback on READMEs, slides, and they can be practice audiences if you ask them nicely!

Assignments Due:

1. Neural Network Exercise - Optional Kaggle Competition
    a. Explain your Model Changes
    b. Recommend you submit an entry!
2. Project 2 - Part 5 - Presentation slides
    a. Remember: NON-DATA SCIENCE AUDIENCE!!!

Daily Schedule

Next Lecture: SQL Queries

Please Read:

1. Intro to SQL
2. SQL Alchemy
3. SELECT and FROM
4. WHERE
5. Wildcards (%), and LIKE
6. Advanced WHERE
7. ORDER BY and LIMIT

Local Environment Installation

If you haven’t already:
Be sure you get this done before the end of the week

DON’T WAIT UNTIL NEXT THURSDAY/FRIDAY

Get help from instructors and TAs during the week on the Environment Install Discord Channel

Required assignment for next week AND
Critical for first week of Data Enrichment