Deeplearning

from zero to hero

Gianluca Carucci | @rucka | gianluca.carucci.org


what I do
Software Engineer | Agile coach
@Hyperfair inc

communities

Ugidotnet | Scala Milano Meetup

Neural Networks for Machine Learning

About this course: Learn about artificial neural networks and how they're being used for machine learning, as applied to speech and object recognition, image segmentation, modeling language and human motion, etc. We'll emphasize both the basic algorithms and the practical tricks needed to get

More

Created by: University of Toronto

UNIVERSITY OF TORONTO

Taught by: Geoffrey Hinton, Professor
Department of Computer Science

how we meet

derivatives of a logistic neur

$$= (1+e^{-z})^{-1}$$

$$\left(\frac{e^{-z}}{(1+e^{-z})^2}\right) = \left(\frac{1}{1+e^{-z}}\right)\left(\frac{e^{-z}}{1+e^{-z}}\right) =$$

$$\frac{}{z} = \frac{(1+e^{-z})-1}{1+e^{-z}} = \frac{(1+e^{-z})}{1+e^{-z}}$$


not a good start


is there another way?


goal


look mum, no code!

How to Retrain Inception's Final Layer for New Categories

transfer learning[1]

Modern object recognition models have millions of parameters and can take weeks fully train. Transfer learning is a technique that shortcuts a lot of this work by taking a fully-trained model for a set of categories like ImageNet, and retrains from the existing weights for new classes. In this example we'll be retraining the final layer from scratch, while leaving all the others untouched. For more information on the approach you can see this paper on Decaf.

Though it's not as good as a full training run, this is surprisingly effective for many applications, and can be run in as little as thirty minutes on a laptop, without requiring a GPU. This tutorial will show you how to run the example script on your own images, and will explain some of the options you have to help control the training process.

1 - How to Retrain Inception's Final Layer for New Categories
- TensorFlow for Poets

Note: A version of this tutorial is also available as a codelab.

Before you start, you must install tensorflow.

the simpsons detector

```
root@cc5a496bf40e:/code/simpson_retraining# ./retrain.sh
INFO:tensorflow:Looking for images in 'marge'
INFO:tensorflow:Looking for images in 'homer'
INFO:tensorflow:Looking for images in 'bart'
INFO:tensorflow:Looking for images in 'lisa'
....
INFO:tensorflow:2018-02-25 09:26:27.689515: Step 0: Train accuracy = 53.0%
INFO:tensorflow:2018-02-25 09:26:27.689750: Step 0: Cross entropy = 1.367082
INFO:tensorflow:2018-02-25 09:26:28.583996: Step 0: Validation accuracy = 33.0% (N=100)
....
INFO:tensorflow:2018-02-25 09:26:49.359180: Step 30: Train accuracy = 65.0%
INFO:tensorflow:2018-02-25 09:26:49.359404: Step 30: Cross entropy = 1.172318
INFO:tensorflow:2018-02-25 09:26:49.871993: Step 30: Validation accuracy = 69.0% (N=100)
....
INFO:tensorflow:2018-02-25 09:35:37.061001: Step 1000: Train accuracy = 95.0%
INFO:tensorflow:2018-02-25 09:35:37.061221: Step 1000: Cross entropy = 0.311776
INFO:tensorflow:2018-02-25 09:35:37.600067: Step 1000: Validation accuracy = 87.0% (N=100)
....
INFO:tensorflow:2018-02-25 09:44:03.139348: Step 2000: Train accuracy = 95.0%
INFO:tensorflow:2018-02-25 09:44:03.139593: Step 2000: Cross entropy = 0.286085
INFO:tensorflow:2018-02-25 09:44:03.538799: Step 2000: Validation accuracy = 83.0% (N=100)
....
INFO:tensorflow:2018-02-25 10:00:39.674607: Step 3999: Train accuracy = 97.0%
INFO:tensorflow:2018-02-25 10:00:39.674859: Step 3999: Cross entropy = 0.175444
INFO:tensorflow:2018-02-25 10:00:40.073648: Step 3999: Validation accuracy = 90.0% (N=100)
INFO:tensorflow:Final test accuracy = 90.4% (N=261)
```

```
root:/code/simpson_retraining# ./evaluate.sh /data/simpson/test_set/0.jpg`
bart 0.63594097
marge 0.3191907
lisa 0.034744244
homer 0.010124116
```


common scenarios

tensorflow / models

Watch 1,966 | Star 28,620 | Fork 15,744

Code | Issues 434 | Pull requests 178 | Projects 2 | Wiki | Insights

Models and examples built with TensorFlow

1,819 commits | 12 branches | 2 releases | 304 contributors | Apache-2.0

Branch: master | New pull request | Create new file | Upload files | Find file | Clone or download

a-dai Merge pull request #3414 from a-dai/master ... Latest commit d903d5e 13 hours ago

official	Merge pull request #3400 from asimshankar/mnist-eager	6 days ago
research	Merge pull request #[illegible]	[illegible]
samples	Fix one more Distributed_T[illegible]	21 [illegible]
tutorials	six.moves works with python2/3	14 days ago
.gitignore	Added PyCharm to .gitignore	6 months ago
.gitmodules	Move the research models into a research subfolder (#2430)	5 months ago
AUTHORS	Spatial Transformer model	2 years ago
CODEOWNERS	Add a-dai to CODEOWNERS for adversarial_text.	6 days ago
CONTRIBUTING.md	Fixing small typo	8 months ago
ISSUE_TEMPLATE.md	Update ISSUE_TEMPLATE.md	15 days ago
LICENSE	Update LICENSE	2 years ago
README.md	Update Readme to link to master research models (#3318)	17 days ago
WORKSPACE	Consolidate privacy/ and differential_privacy/.	a year ago

README.md

TensorFlow Models

This repository contains a number of different models implemented in TensorFlow:

The official models are a collection of example models that use TensorFlow's high-level APIs. They are intended to be well-maintained, tested, and kept up to date with the latest stable TensorFlow API. They should also be reasonably optimized for fast performance while still being easy to read. We especially recommend newer TensorFlow users to start here.

The research models are a large collection of models implemented in TensorFlow by researchers. They are not officially supported or available in release branches; it is up to the individual researchers to maintain the models and/or provide support on issues and pull requests.

The samples folder contains code snippets and smaller models that demonstrate features of

common solutions

ready to go models[2]

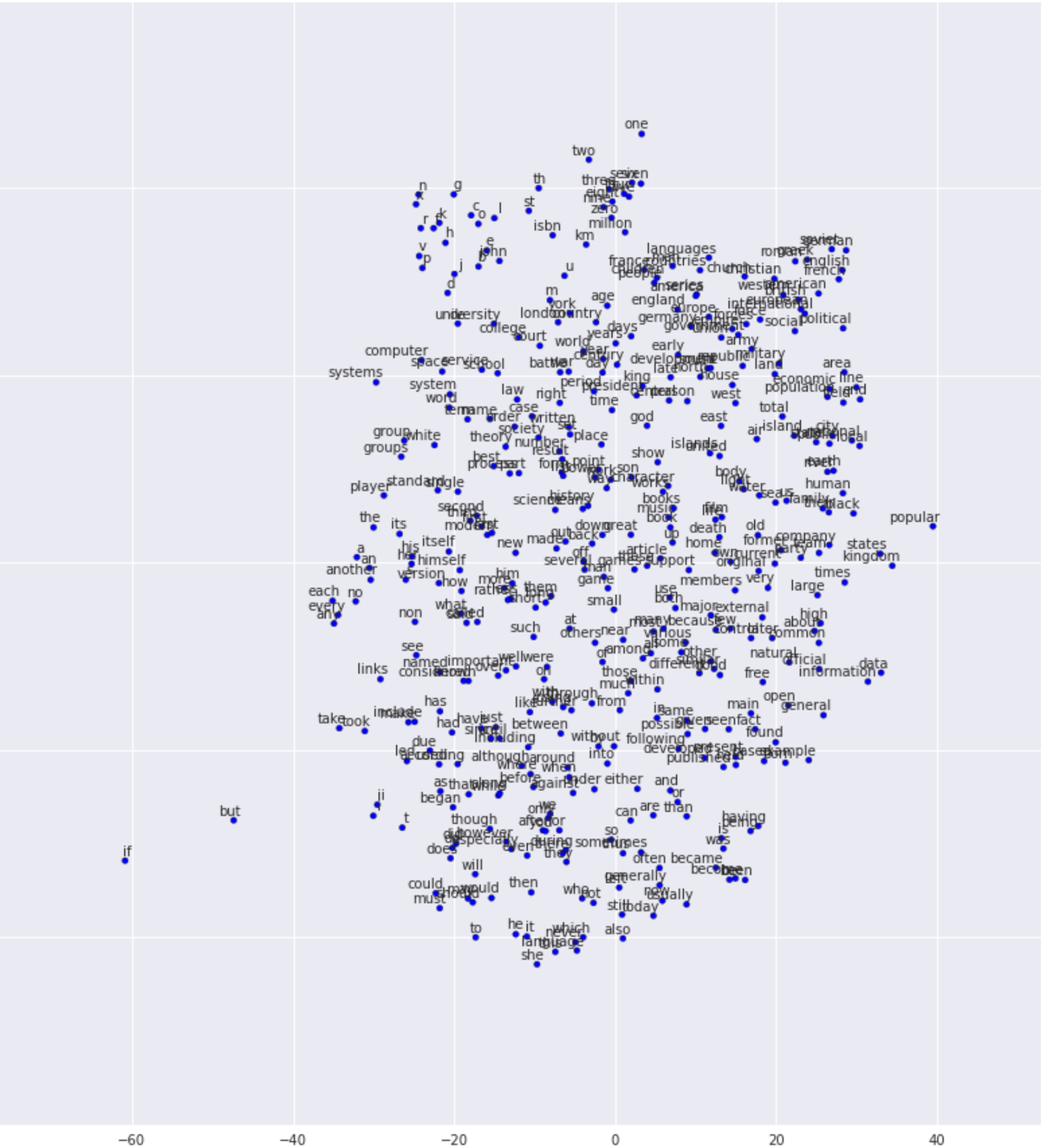
[2] TensorFlow models

- audioset: Models and supporting code for use with AudioSet.
- autoencoder: various autoencoders.
- brain_coder: Program synthesis with reinforcement learning.
- cognitive_mapping_and_planning: implementation of a spatial memory based mapping and planning architecture for visual navigation.
- compression: compressing and decompressing images using a pre-trained Residual GRU network.
- delf: deep local features for image matching and retrieval.
- differential_privacy: privacy-preserving student models from multiple teachers.
- domain_adaptation: domain separation networks.
- gan: g[illegible]nerative adv[illegible]rsari[illegible] networks.
- [illegible]: [illegible]ag[illegible]captioning.
- inc[illegible]: [illegible] for comp[illegible]er vision.
- learning_to_remember_rare_events: a large-scale life-long memory module for use in deep learning.
- lfads: sequential variational autoencoder for analyzing neuroscience data.
- lm_1b: language modeling on the one billion word benchmark.
- namignizer: recognize and generate names.
- neural_gpu: highly parallel neural [illegible]puter.
- neural_programmer: neural network augmented with logic and mathematic operations.
- nex[illegible] prob[illegible]stic future frame synthesis via cross convolutional networks.
- object_detection: localizing and identifying multiple objects in a single image.
- pcl_rl: code for several reinforcement learning algorithms, including Path Consistency Learning.
- ptn: perspective transformer nets for 3D object reconstruction.
- qa_kg: module networks for question answering on knowledge graphs.
- real_nvp: density estimation using real-valued non-volume preserving (real NVP) transformations.
- rebar: low-variance, unbiased gradient estimates for discrete latent variable models.
- resnet: deep and wide residual networks.
- skip_thoughts: recurrent neural network sentence-to-vector encoder.
- slim: image classification models in TF-Slim.
- street: identify the name of a street (in France) from an image using a Deep RNN.
- swivel: the Swivel algorithm for generating word embeddings.
- syntaxnet: neural models of natural language syntax.
- tcn: Self-supervised representation learning from multi-view video.

common solutions

ready to go libraries[3]

[3] - OpenCV-Python
- Spacy
- gensim



Word Algebra[4]

```
from gensim.models.KeyedVectors import load_word2vec_format

model = load_word2vec_format(
    'GoogleNews-vectors-negative300.bin',
    limit=2000000, binary=True)

#calculate: (Rome - Italy) + France = ?
model.similar_by_vector(
    model['Rome'] - model['Italy'] + model['France']
    , topn=1)
#[(u'Paris', 0.7167540192604065)]

#calculate: (doctor - man) + woman = ?
model.most_similar(
    model['doctor'] - model['man'] + model['woman']
    , topn=1)
#[(u'nurse', 0.7127888798713684)]

#calculate: (gone - go) + eat = ?
model.most_similar(
    positive=['gone', 'eat'], negative=['go']
    , topn=1)
#[(u'eaten', 0.7462186217308044)]
```

[4] Credits: The dark side of deep learning - Simone Scardapane - Codemotion 2017

Datasets

1. MNIST Handwritten digits
2. Google House Numbers from street view
3. CIFAR-10 and CIFAR-100
4. IMAGENET
5. Tiny Images 80 Million tiny images6.
6. Flickr Data 100 Million Yahoo dataset
7. Berkeley Segmentation Dataset 500
8. UC Irvine Machine Learning Repository
9. Flickr 8k
10. Flickr 30k
11. Microsoft COCO
12. VQA
13. [illegible]
14. [illegible]
15. AVHRR Pathfinder
16. Air Freight - The Air Freight data set is a ray-traced image sequence along with ground truth segmentation based on

common solutions

ready to go dataset[5]

[5] - Awesome deep learning#datasets (Github)
- List of datasets for machine learning research (Wikipedia)
- Deep Learning datasets


RECIPES
what else?
RECIPE
Pork Chalupas
from Grams
cooking time 6-8 hours
serves many
2-3# pork roast
preheat oven to crock pot
(I remove fat after roast begins to be
tender or you can do it before cooked)
or stovetop
1 lb. pinto beans
cloves garlic
chopped
1 cube beef bouillon
or canned broth
1 jalapeno (opt.)
OVER

ingredients

keras

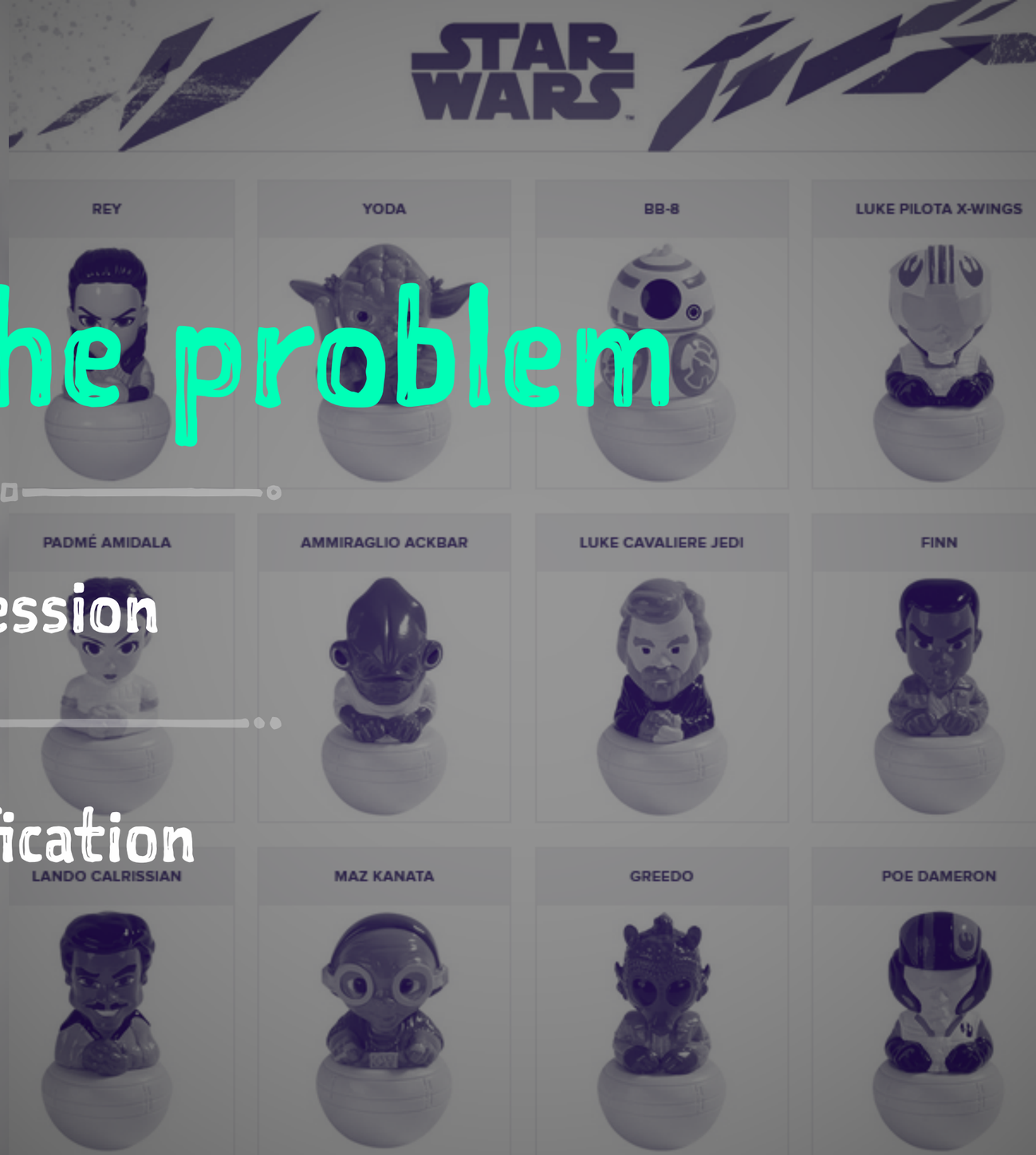
tensorflow

deep learning in 5 steps

1. define the problem

regression

classification



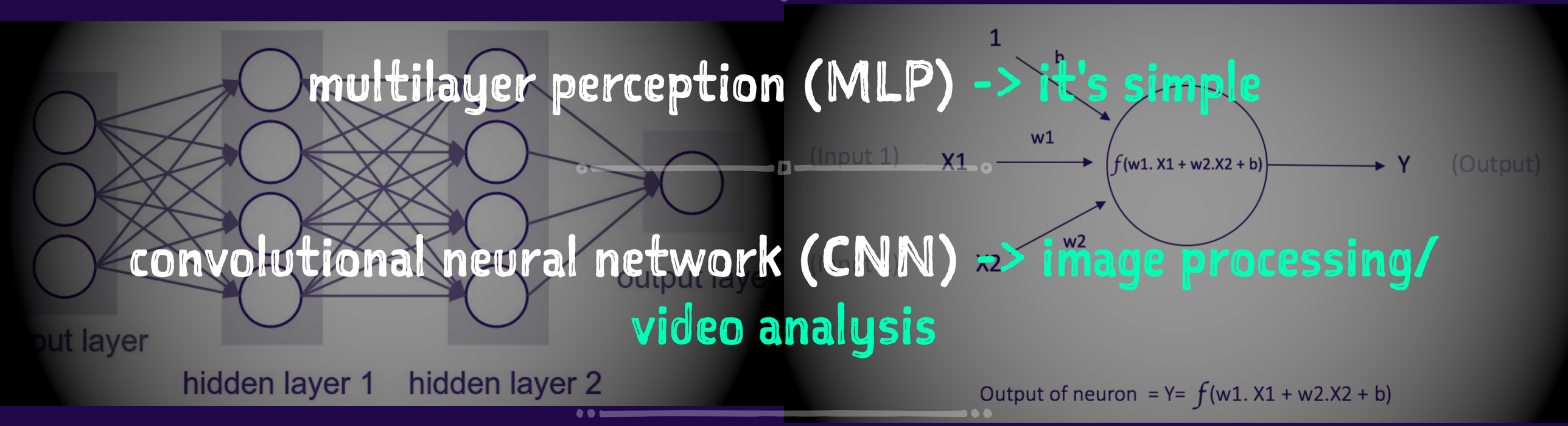
2. prepare your data

choose/create dataset

clean data

normalize data

3. define the model



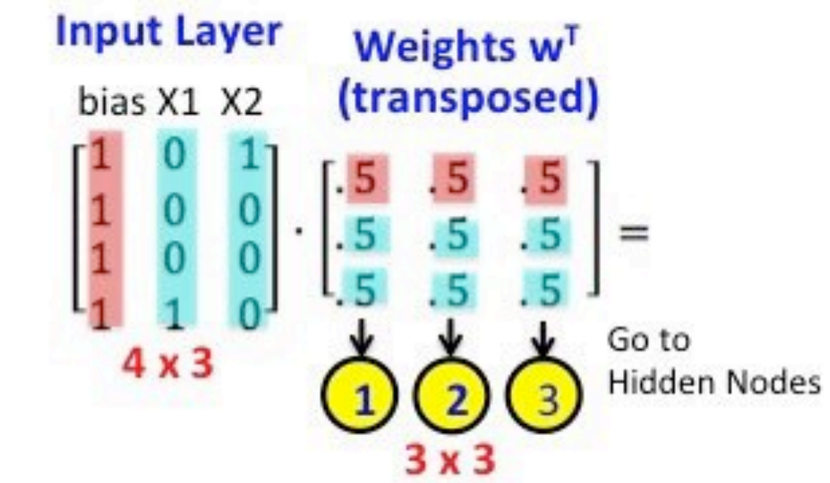
multilayer perception (MLP) -> it's simple

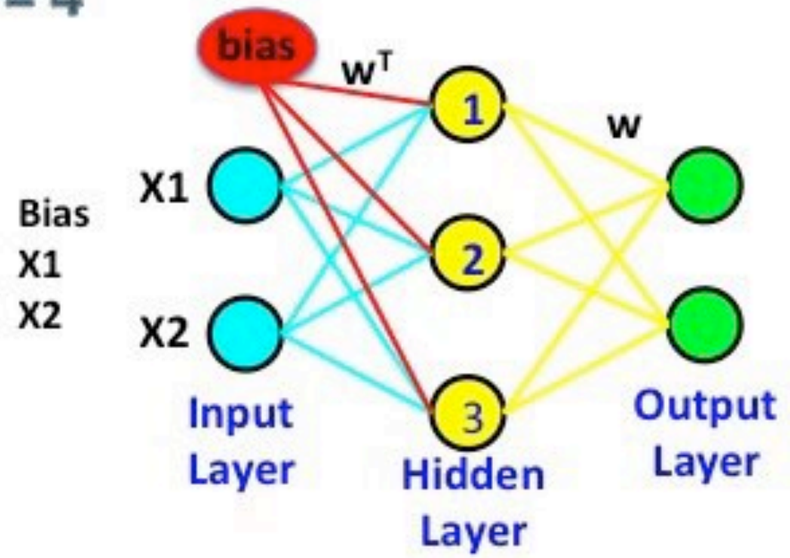
convolutional neural network (CNN) -> image processing/ video analysis

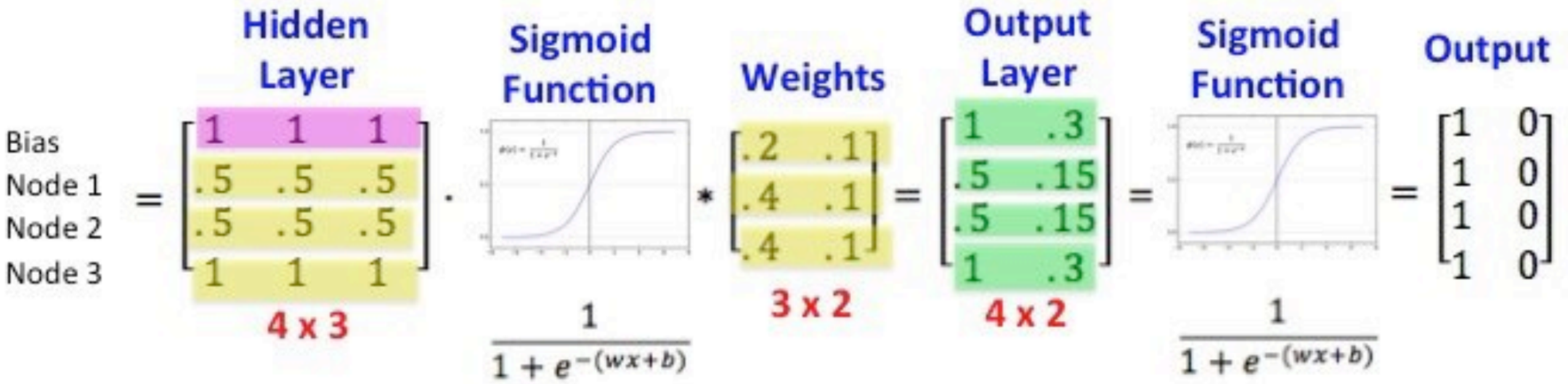
long short-term memory (LSTM) -> next sequence prediction

Neural Networks

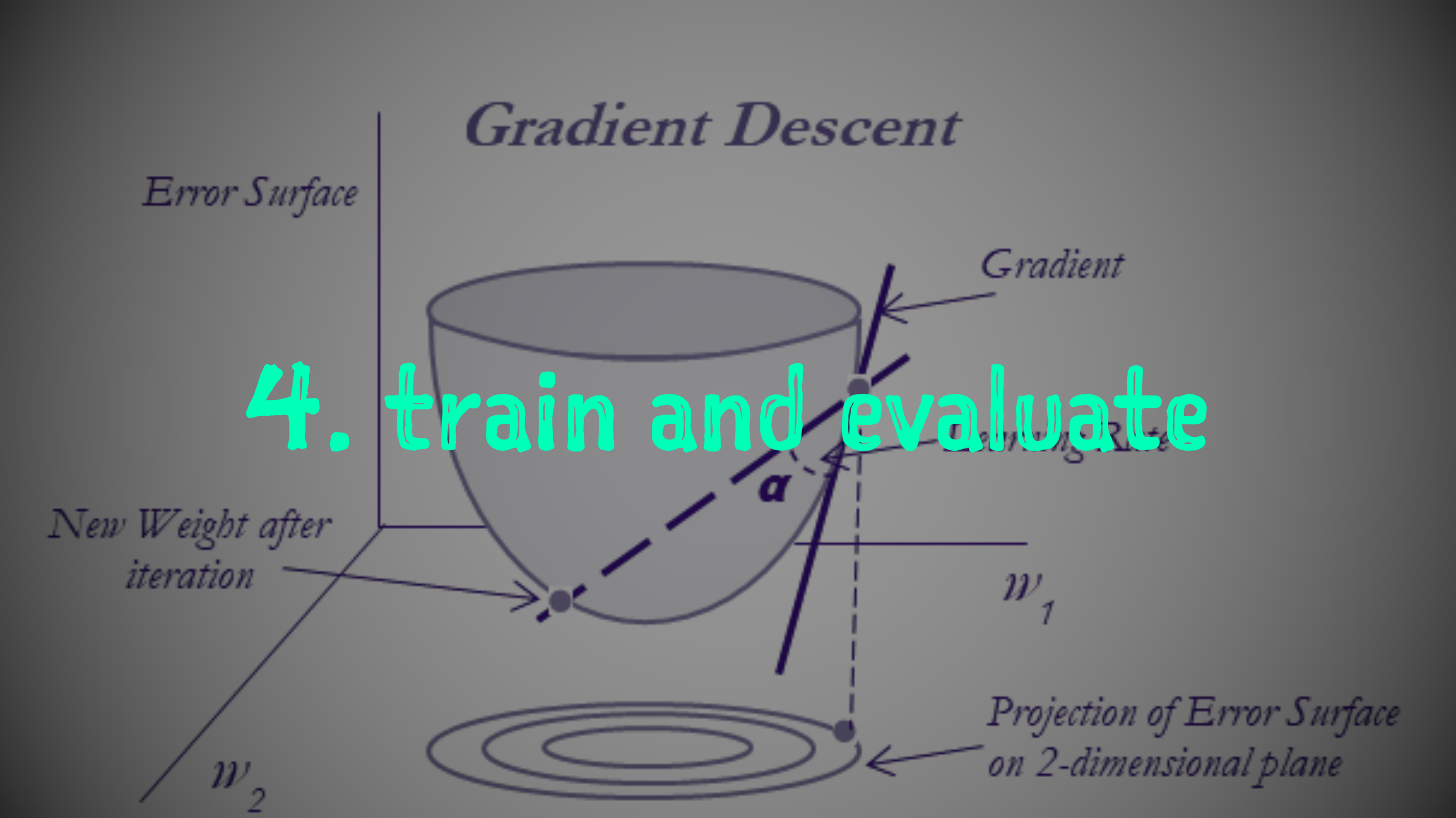
Color Guided Matrix Multiplication for a Binary Classification Task with N = 4





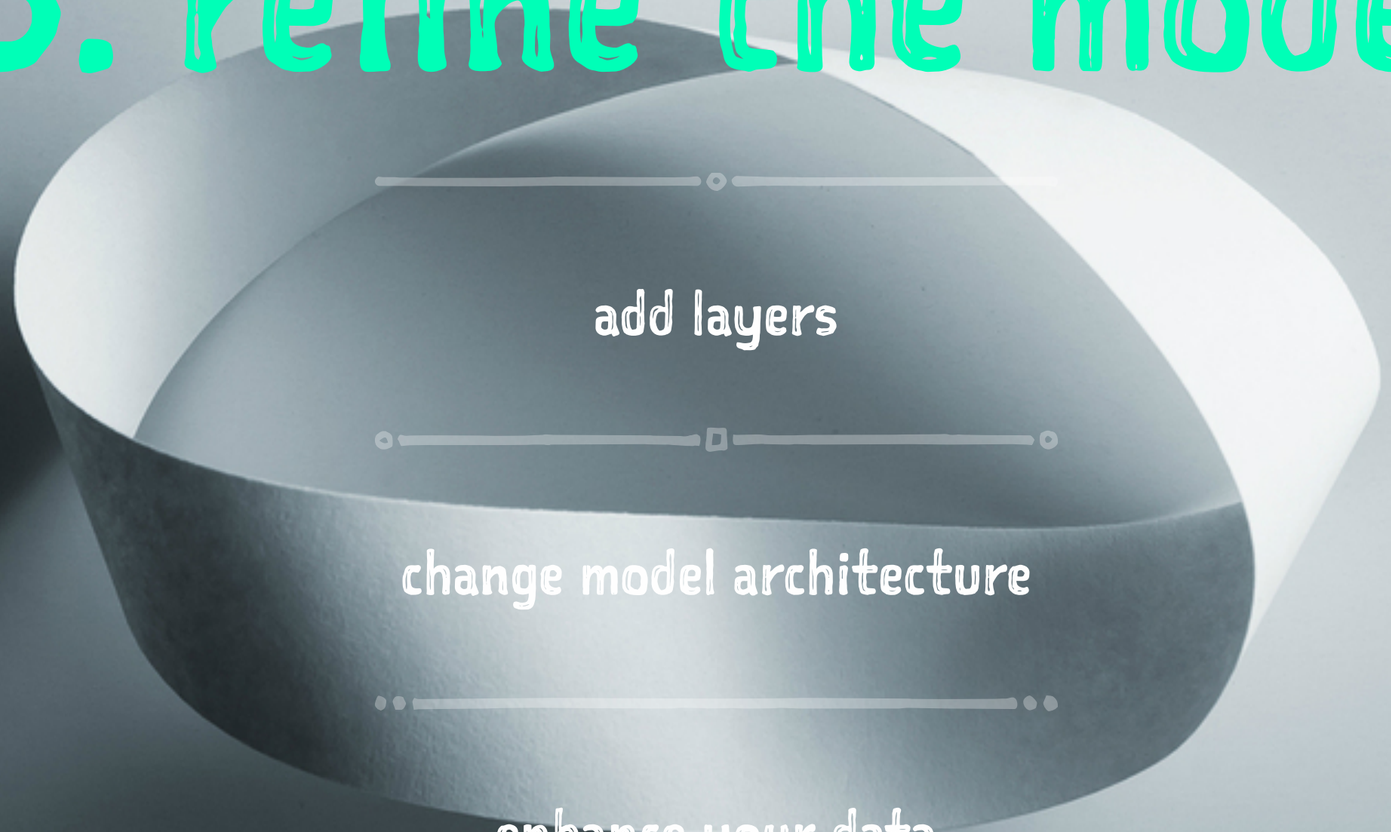


Rubens Zimbres



4. train and evaluate

5. refine the model



add layers

change model architecture

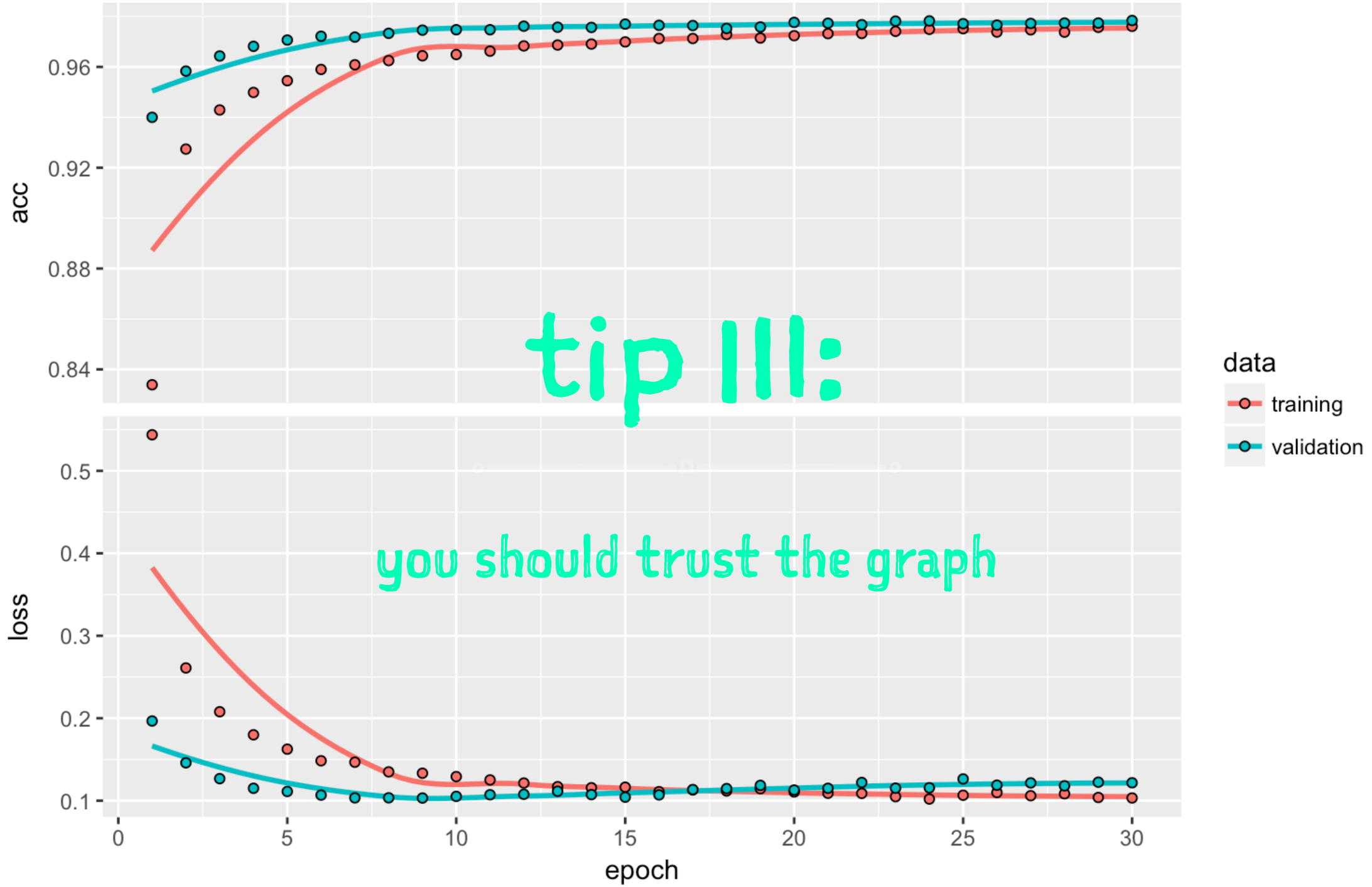
enhance your data

tip 1:

your dataset is the key

please, start with the simplest model possible
please, start with the simplest model possible
please, start with the simplest model possible
please, start with the simplest model possible
please, start with the simplest model possible
please, start with the simplest model possible
please, start with the simplest model possible
please, start with the simplest model possible
please, start with the simplest model possible
please, start with the simplest mo

tip 11:


acc
0.96
0.92
0.88
0.84
loss
0.5
0.4
0.3
0.2
0.1
0
5
10
15
20
25
30
epoch
data
training
validation
tip III:
you should trust the graph

tip IV:

patient

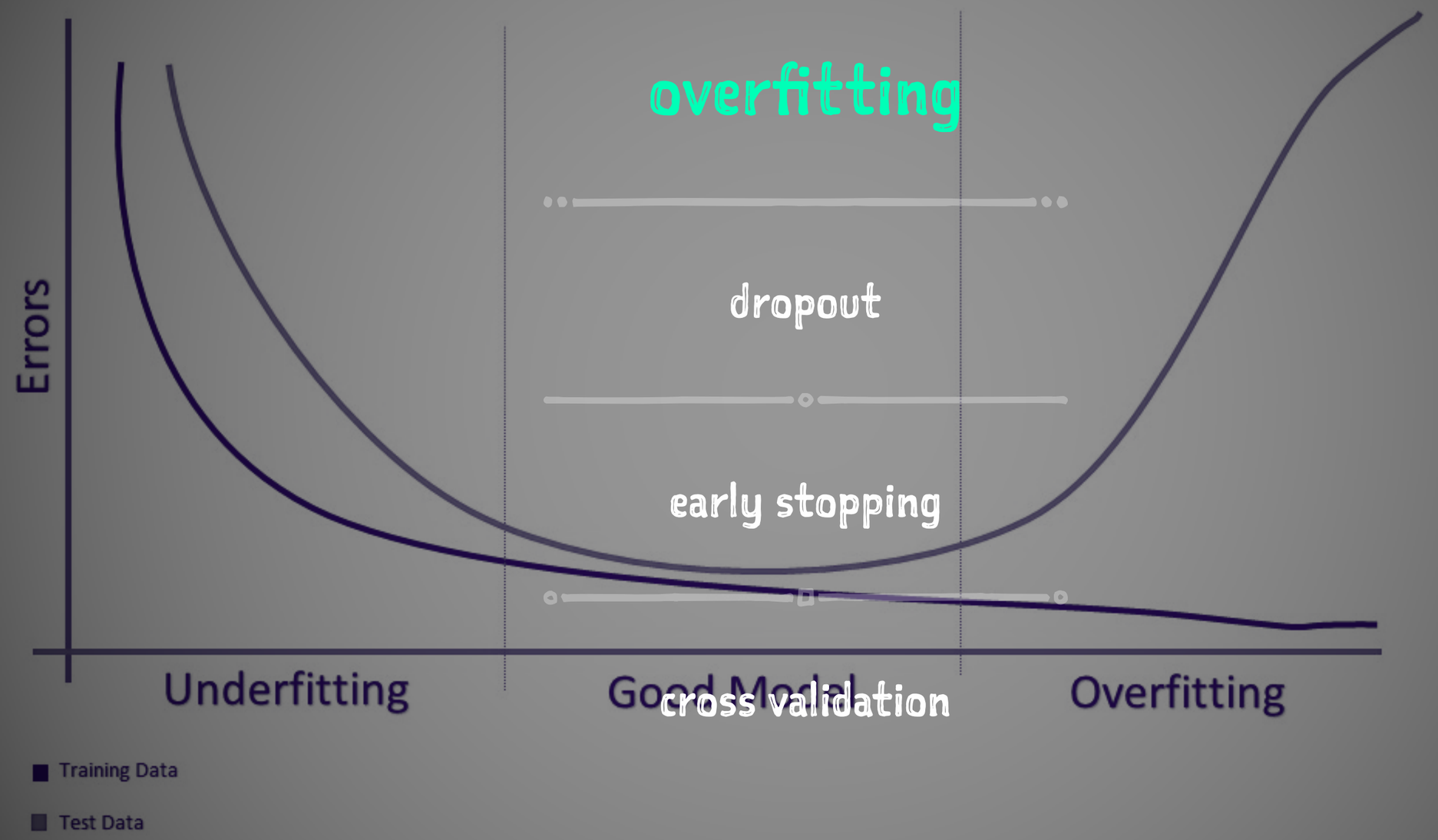
bonus track: hyperparameters tuning[6]

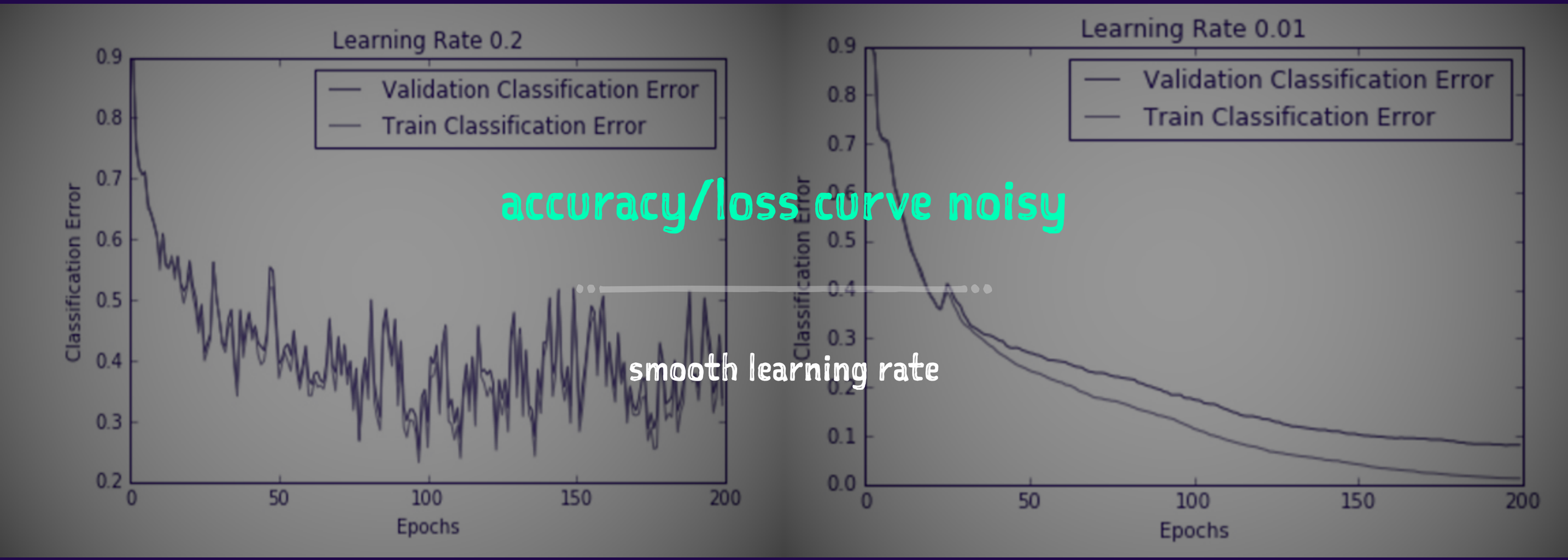
scikit learn GridSearchCV, RandomizedSearchCV

hyperopt

[6] - scikit learn Exhaustive&Random Grid Search
- Hyperopt


troubleshooting


overfitting
dropout
early stopping
cross validation
Errors
Underfitting
Good Model
Overfitting
Training Data
Test Data


Learning Rate 0.2
Validation Classification Error
Train Classification Error
Classification Error
Epochs
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0
50
100
150
200
accuracy/loss curve noisy
smooth learning rate
Learning Rate 0.01
Validation Classification Error
Train Classification Error
Classification Error
Epochs
0.9
0.8
0.7
0.5
0.4
0.3
0.1
0.0
0
50
100
150
200

Resources #1

- Google Machine Learning Crash Course
- Learn TensorFlow and deep learning, without a Ph.D. by Martin Görner
- How to Use Metrics for Deep Learning with Keras in Python
- Neural Networks for Machine Learning course from Coursera

Resources #2

- Machine learning finance
- Keras
- Tensorflow
- rucka/deeplearning docker image

It is a capital mistake to theorize before one has data

Sherlock Holmes


questions?


thank you!
slide & code
https://github.com/rucka/pycon9
https://github.com/rucka/deeplearning_docker