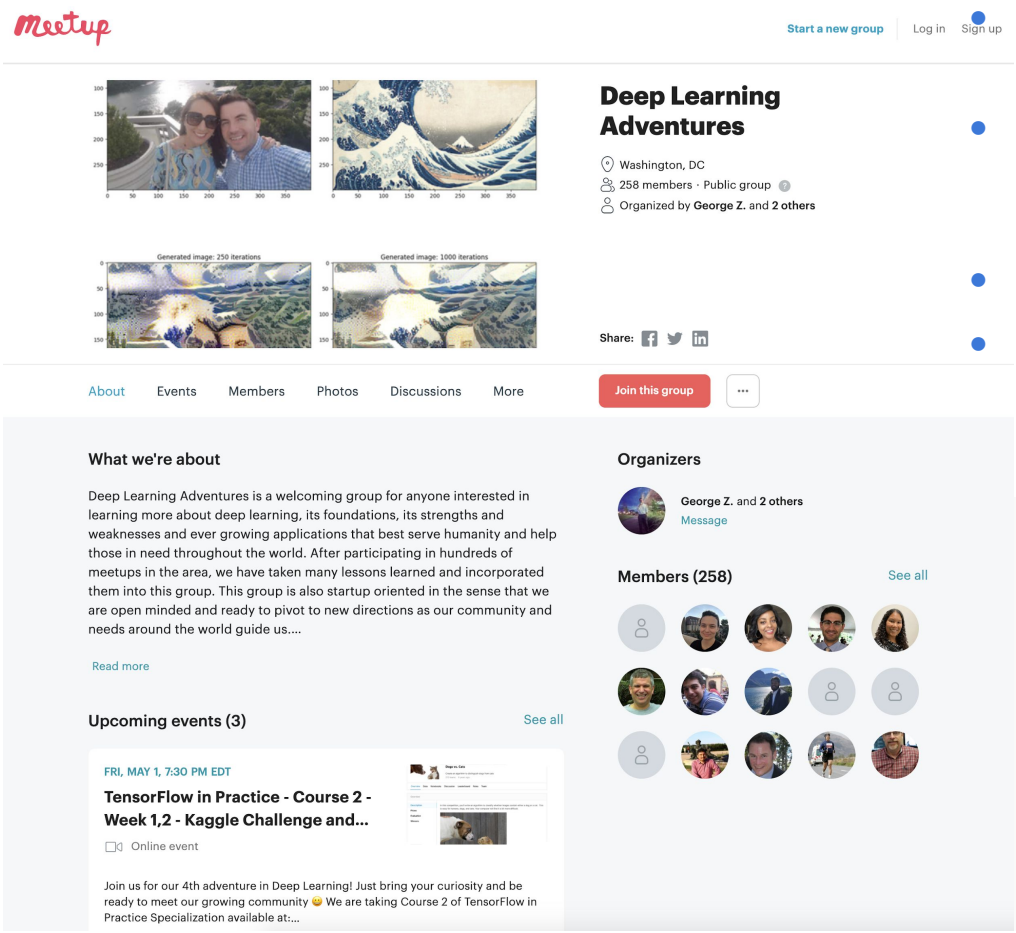
Deep Learning Adventures TensorFlow In Practice - Presentation 3

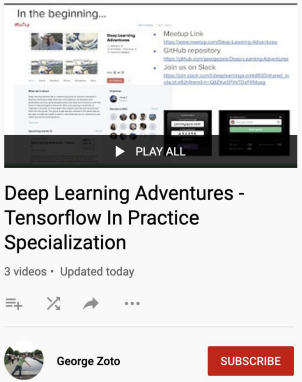
A quick overview of Coursera's Tensorflow in Practice specialization course

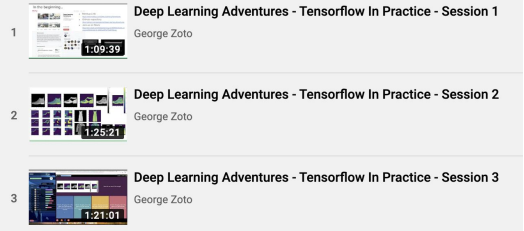
Robert Kraig, David Patton, George Zoto
https://www.meetup.com/Deep-Learning-Adventures

In the beginning...

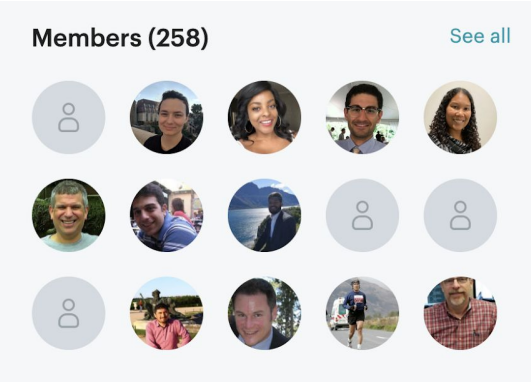


- Meetup Link
  https://www.meetup.com/Deep-Learning-Adventures
- GitHub repository
  https://github.com/georgezoto/Deep-Learning-Adventures
- Join us on Slack
  https://join.slack.com/t/deeplearninga-nmk8930/shared_invite/zt-d52h9mm9-h~Q0ZXw5PXsTDzPIINIvog
- Need a refresher or new to Deep Learning?
- YouTube recordings of all our Meetups 😀
  https://bit.ly/deep-learning-tf





Get to know our community



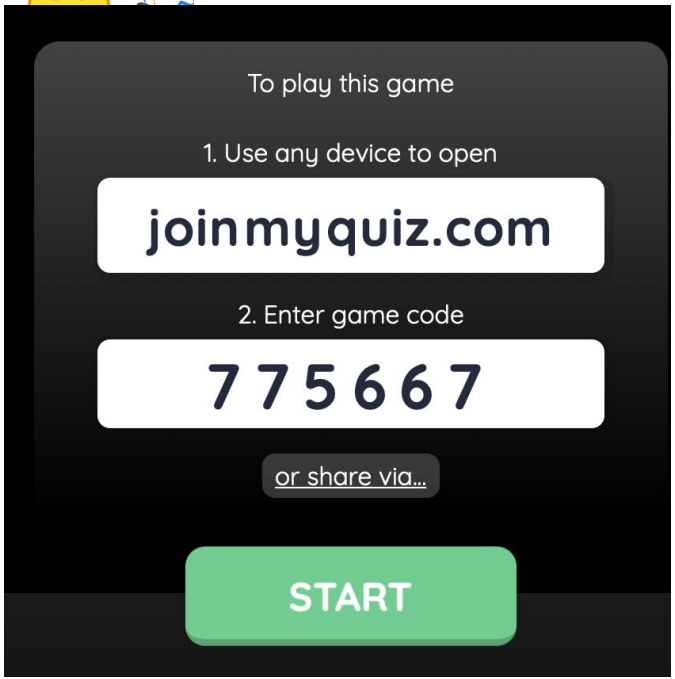
Hello, my name is ____ and this is my ____ meetup.

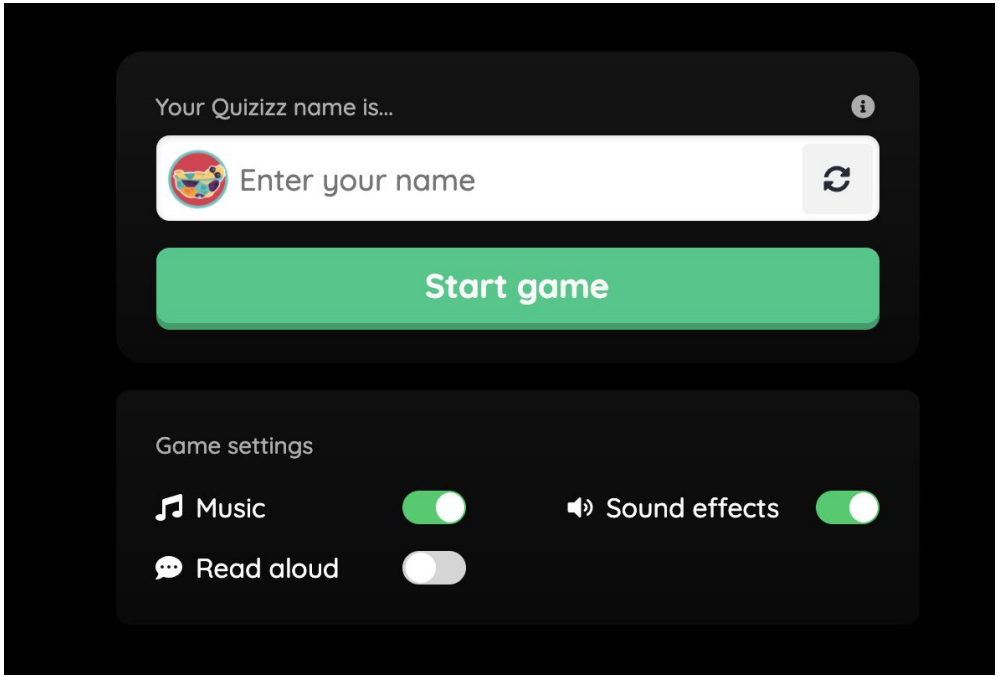
I enjoy ____ and I am interested in learning more about ____

Training set 😀

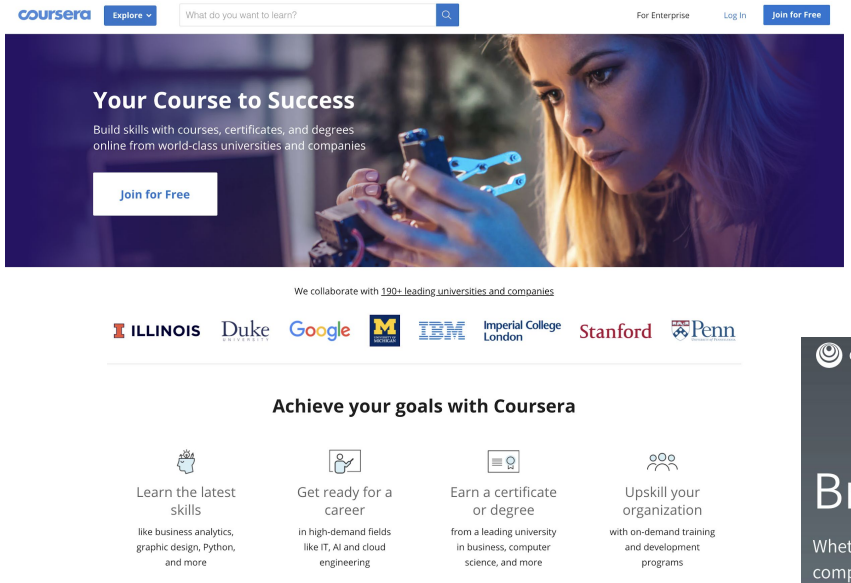
Hello, my name is **George** and this is my **4th** meetup. I enjoy **applying deep learning to solve interesting problems** and I am interested in learning more about **data augmentation, transfer learning and NLP**.

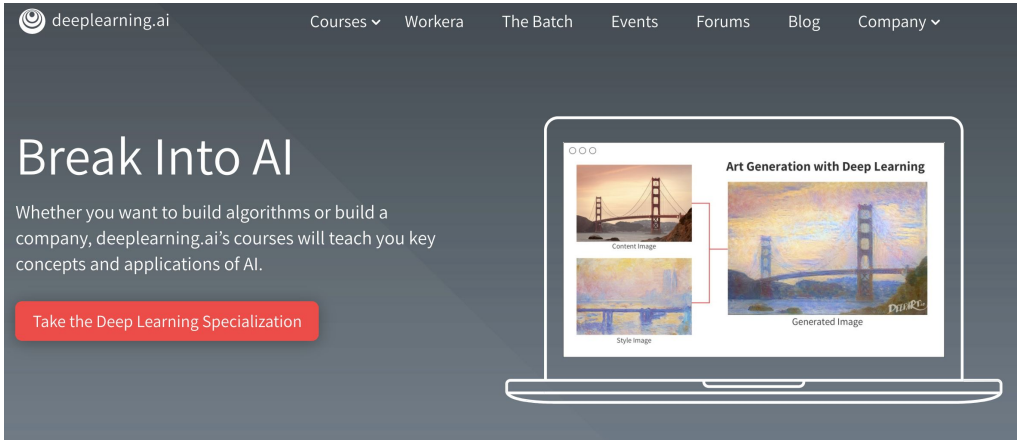
Not a typical Meetup... Get ready for a fun game 🥳





Attribution to Coursera and deeplearning.ai





Source:
https://www.coursera.org/about/terms
https://www.coursera.org/
https://www.deeplearning.ai/

Chapter 1 - TensorFlow in Practice Specialization

About this Specialization

199,621 recent views

Discover the tools software developers use to build scalable AI-powered algorithms in TensorFlow, a popular open-source machine learning framework.

In this four-course Specialization, you'll explore exciting opportunities for AI applications. Begin by developing an understanding of how to build and train neural networks. Improve a network's performance using convolutions as you train it to identify real-world images. You'll teach machines to understand, analyze, and respond to human speech with natural language processing systems. Learn to process text, represent sentences as vectors, and input data to a neural network. You'll even train an AI to create original poetry!

AI is already transforming industries across the world. After finishing this Specialization, you'll be able to apply your new TensorFlow skills to a wide range of problems and projects.

Looking for more advanced TensorFlow content? Check out the new **TensorFlow: Data and Deployment Specialization**.

Source: https://www.coursera.org/specializations/tensorflow-in-practice

Chapter 1 - TensorFlow in Practice Specialization

There are 4 Courses in this Specialization

COURSE 1

Introduction to TensorFlow for Artificial Intelligence, Machine Learning, and Deep Learning

★★★★½ 4.7 6,196 ratings • 1,282 reviews

If you are a software developer who wants to build scalable AI-powered algorithms, you need to understand how to use the tools to build them. This course is part of the upcoming Machine Learning in Tensorflow Specialization and will teach you best practices for using TensorFlow, a popular open-source framework for machine learning.

SHOW ALL

COURSE 2

Convolutional Neural Networks in TensorFlow

★★★★½ 4.7 2,751 ratings • 410 reviews

If you are a software developer who wants to build scalable AI-powered algorithms, you need to understand how to use the tools to build them. This course is part of the upcoming Machine Learning in Tensorflow Specialization and will teach you best practices for using TensorFlow, a popular open-source framework for machine learning.

SHOW ALL

COURSE 3

Natural Language Processing in TensorFlow

★★★★½ 4.6 2,037 ratings • 277 reviews

If you are a software developer who wants to build scalable AI-powered algorithms, you need to understand how to use the tools to build them. This Specialization will teach you best practices for using TensorFlow, a popular open-source framework for machine learning.

SHOW ALL

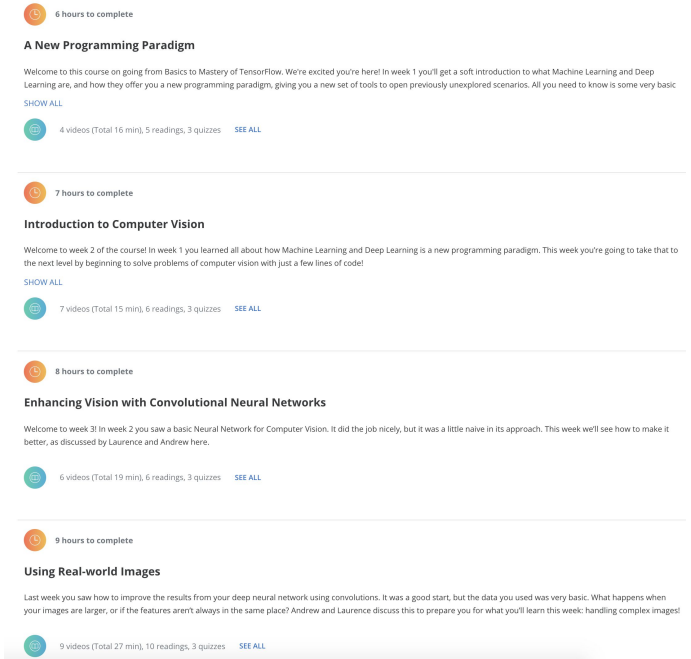
COURSE 4

Sequences, Time Series and Prediction

★★★★½ 4.6 1,374 ratings • 223 reviews

If you are a software developer who wants to build scalable AI-powered algorithms, you need to understand how to use the tools to build them. This Specialization will teach you best practices for using TensorFlow, a popular open-source framework for machine learning.

SHOW ALL



Source: https://www.coursera.org/specializations/tensorflow-in-practice

Setup

Colaboratory is a free Jupyter notebook environment that requires no setup and runs entirely in the cloud. You can write and execute code, save and share your analyses, and access powerful computing resources, all for free from your browser.

https://colab.research.google.com

Course 2: Convolutional Neural Networks in TensorFlow

Week 1: Exploring a Larger Dataset
Week 2: Augmentation: A technique to avoid overfitting
Week 3: Transfer Learning
Week 4: Multiclass Classifications

Exploring a Larger Dataset

Dogs vs. Cats

Create an algorithm to distinguish dogs from cats

215 teams · 6 years ago

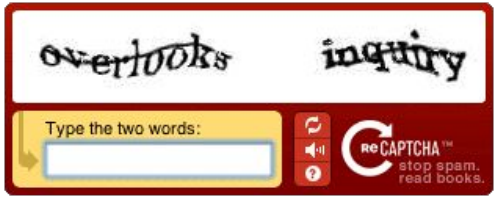
Overview Data Notebooks Discussion Leaderboard Rules Team

Overview

Description
Prizes
Evaluation
Winners

In this competition, you'll write an algorithm to classify whether images contain either a dog or a cat. is easy for humans, dogs, and cats. Your computer will find it a bit more difficult.

Deep Blue beat Kasparov at chess in 1997.
Watson beat the brightest trivia minds at Jeopardy in 2011.
Can you tell Fido from Mittens in 2013?



Human-Interactive Proof: $P(\text{breach}) = p^n$

Kaggle Competition: Dogs vs Cats

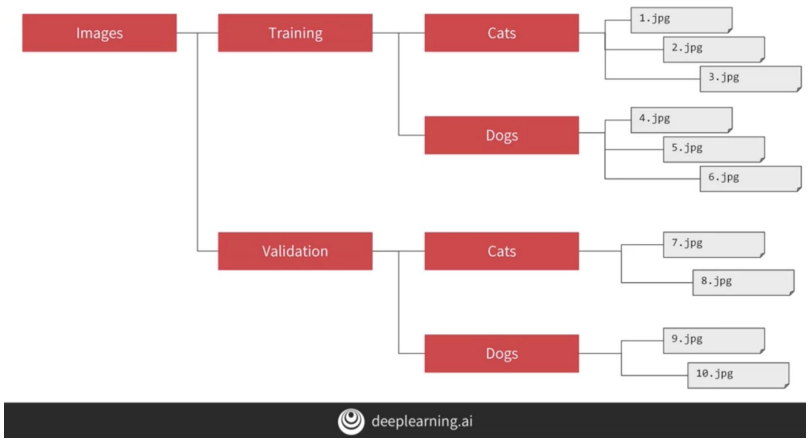
https://www.kaggle.com/c/dogs-vs-cats/
Metric: Binary Classification Accuracy (i.e. no partial credit)

Winner: Pierre Sermanet ---> Overfeat

https://cilvr.nyu.edu/doku.php?id=software:overfeat:start
https://arxiv.org/abs/1312.6229
https://www.youtube.com/watch?v=3U-bZgKFS7g

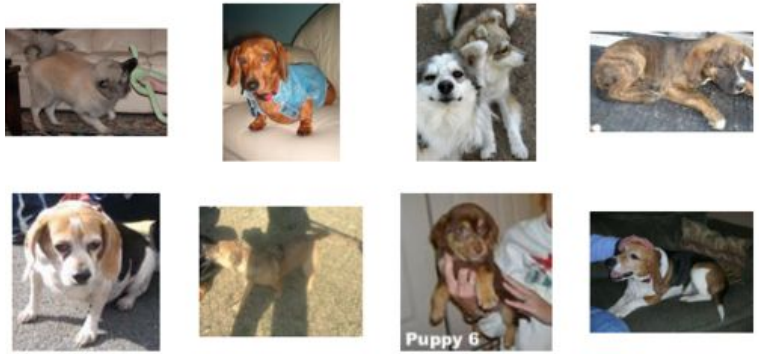
#	△pub	Team Name	Notebook	Team Members	Score	Entries	Last
1	—	Pierre Sermanet			0.98914	5	6y
2	▲4	orchid			0.98308	17	6y
3	—	Owen			0.98171	15	6y
4	—	Paul Covington			0.98171	3	6y
5	▼3	Maxim Milakov			0.98137	24	6y
6	▼1	we've been in KAIST			0.98102	8	6y
7	▲1	Doug Koch			0.98057	6	6y
8	▲2	fastml.com/cats-and-dogs			0.98000	6	6y

Exploring a Larger Dataset




```
from tensorflow.keras.preprocessing.image import ImageDataGenerator

# All images will be rescaled by 1./255.
train_datagen = ImageDataGenerator( rescale = 1.0/255. )
test_datagen  = ImageDataGenerator( rescale = 1.0/255. )

# --------------------
# Flow training images in batches of 20 using train_datagen generator
# --------------------
train_generator = train_datagen.flow_from_directory(train_dir,
                                                    batch_size=20,
                                                    class_mode='binary',
                                                    target_size=(150, 150))
# --------------------
# Flow validation images in batches of 20 using test_datagen generator
# --------------------
validation_generator =  test_datagen.flow_from_directory(validation_dir,
                                                         batch_size=20,
                                                         class_mode  = 'binary',
                                                         target_size = (150, 150))
```

```
Found 2000 images belonging to 2 classes.
Found 1000 images belonging to 2 classes.
```



Exploring a Larger Dataset

```
model = tf.keras.models.Sequential([
    # Note the input shape is the desired size of the image 150x150 with 3 bytes color
    tf.keras.layers.Conv2D(16, (3,3), activation='relu', input_shape=(150, 150, 3)),
    tf.keras.layers.MaxPooling2D(2,2),
    tf.keras.layers.Conv2D(32, (3,3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2,2),
    tf.keras.layers.Conv2D(64, (3,3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2,2),
    # Flatten the results to feed into a DNN
    tf.keras.layers.Flatten(),
    # 512 neuron hidden layer
    tf.keras.layers.Dense(512, activation='relu'),
    # Only 1 output neuron. It will contain a value from 0-1,
    # where 0 for 1 class ('cats') and 1 for the other ('dogs')
    tf.keras.layers.Dense(1, activation='sigmoid')
])
```

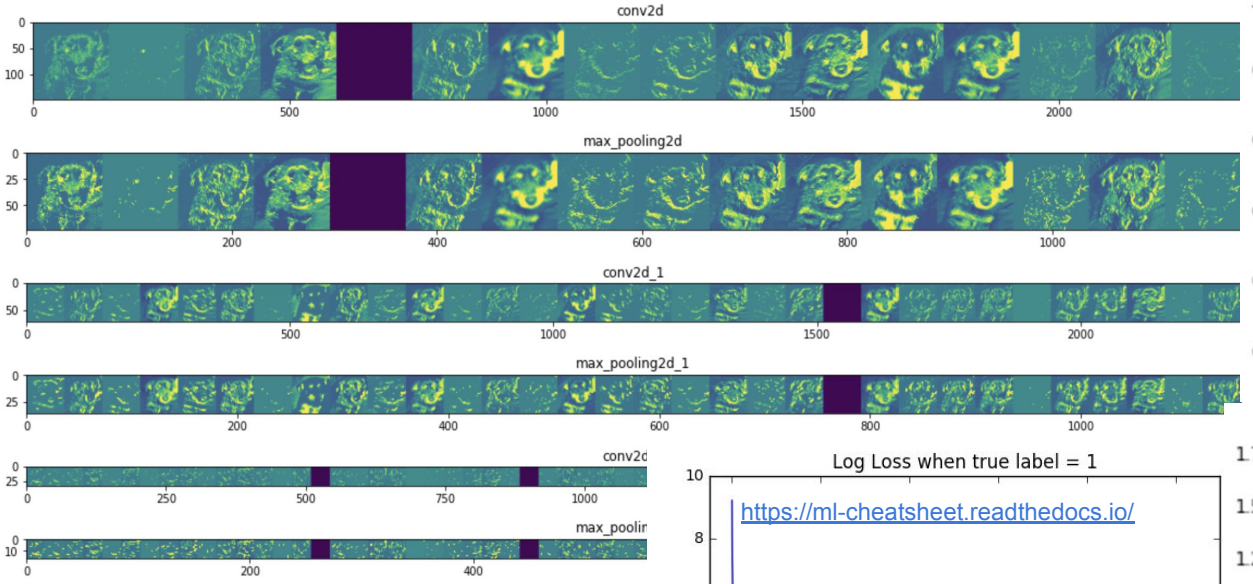
```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 148, 148, 16)      448
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 74, 74, 16)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 72, 72, 32)        4640
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 36, 36, 32)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 34, 34, 64)        18496
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 17, 17, 64)        0
_________________________________________________________________
flatten (Flatten)            (None, 18496)             0
_________________________________________________________________
dense (Dense)                (None, 512)               9470464
_________________________________________________________________
dense_1 (Dense)              (None, 1)                 513
=================================================================
Total params: 9,494,561
Trainable params: 9,494,561
Non-trainable params: 0
```

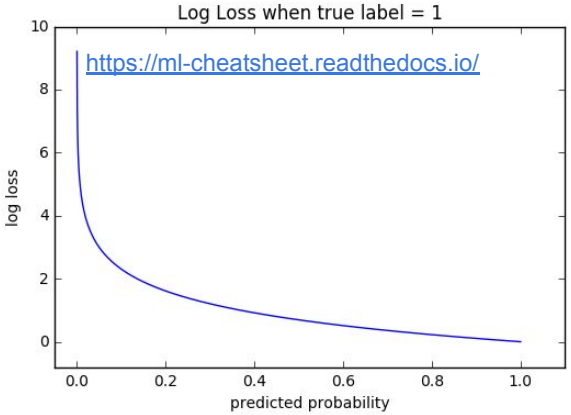
```
from tensorflow.keras.optimizers import RMSprop

model.compile(optimizer=RMSprop(lr=0.001),
              loss='binary_crossentropy',
              metrics = ['accuracy'])
```

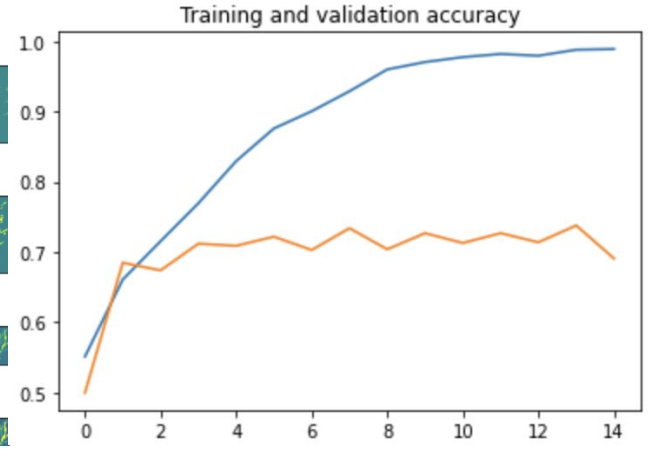
```
history = model.fit(train_generator,
                                validation_data=validation_generator,
                                steps_per_epoch=100,
                                epochs=15,
                                validation_steps=50,
                                verbose=2)
```

Exploring a Larger Dataset



Bug fix: [layer.output for layer in model.layers[1:]]





Model is badly overfitting!

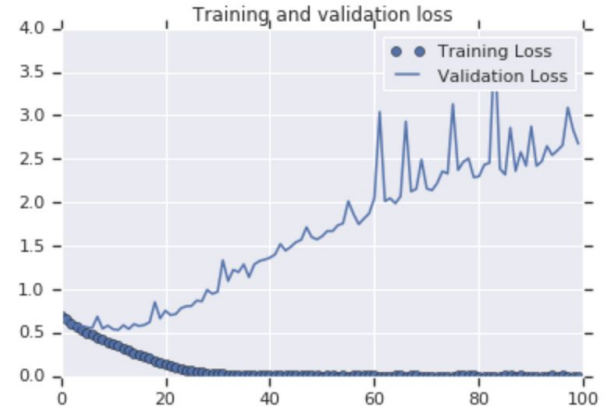
Training Loss Decreasing
Validation Loss Increasing

Course 2: Convolutional Neural Networks in TensorFlow

Week 1: Exploring a Larger Dataset
Week 2: Augmentation: A technique to avoid overfitting
Week 3: Transfer Learning
Week 4: Multiclass Classifications

Augmentation: A technique to avoid overfitting

```
model = tf.keras.models.Sequential([
    tf.keras.layers.Conv2D(32, (3,3), activation='relu', input_shape=(150, 150, 3)),
    tf.keras.layers.MaxPooling2D(2, 2),
    tf.keras.layers.Conv2D(64, (3,3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2,2),
    tf.keras.layers.Conv2D(128, (3,3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2,2),
    tf.keras.layers.Conv2D(128, (3,3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2,2),
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(512, activation='relu'),
    tf.keras.layers.Dense(1, activation='sigmoid')
])

model.compile(loss='binary_crossentropy',
              optimizer=RMSprop(lr=1e-4),
              metrics=['accuracy'])

# All images will be rescaled by 1./255
train_datagen = ImageDataGenerator(rescale=1./255)
test_datagen = ImageDataGenerator(rescale=1./255)
```

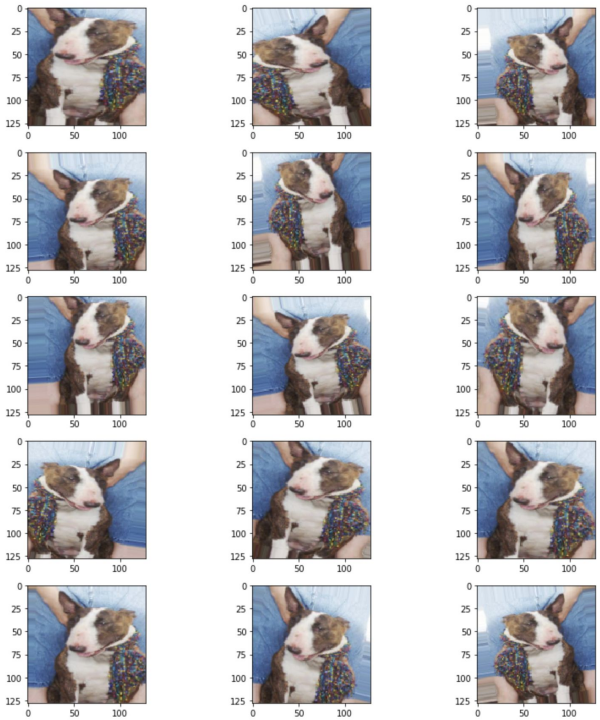


Augmentation: A technique to avoid overfitting

```
# Updated to do image augmentation
train_datagen = ImageDataGenerator(
      rotation_range=40,
      width_shift_range=0.2,
      height_shift_range=0.2,
      shear_range=0.2,
      zoom_range=0.2,
      horizontal_flip=True,
      fill_mode='nearest')
```

```
train_datagen = ImageDataGenerator(
    rotation_range=15,
    rescale=1./255,
    shear_range=0.1,
    zoom_range=0.2,
    horizontal_flip=True,
    width_shift_range=0.1,
    height_shift_range=0.1
)
```

Source: https://www.kaggle.com/uysimty/keras-cnn-dog-or-cat-classification

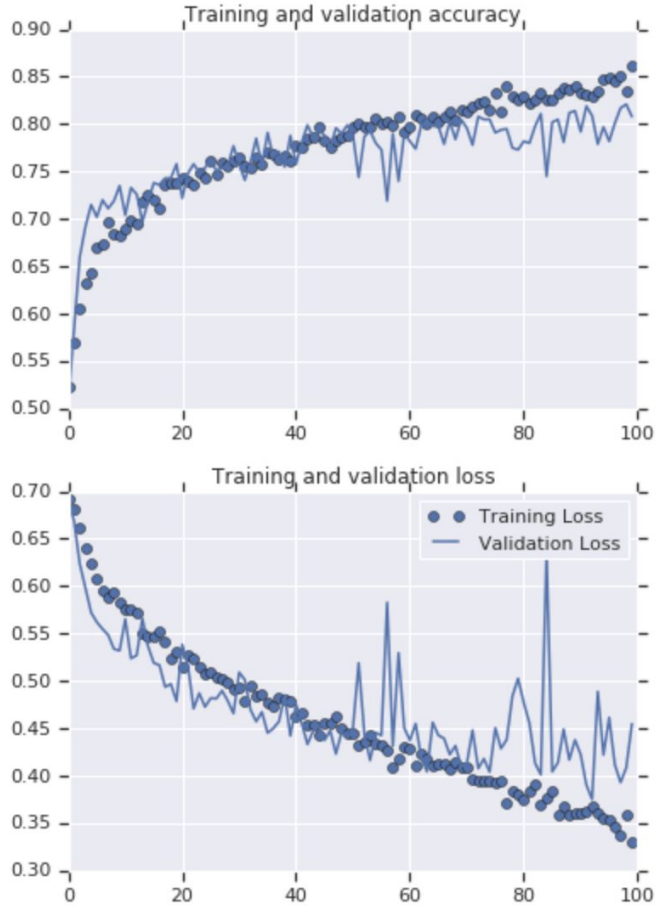


These are just a few of the options available (for more, see the Keras documentation. Let's quickly go over what we just wrote:

- rotation_range is a value in degrees (0–180), a range within which to randomly rotate pictures.
- width_shift and height_shift are ranges (as a fraction of total width or height) within which to randomly translate pictures vertically or horizontally.
- shear_range is for randomly applying shearing transformations.
- zoom_range is for randomly zooming inside pictures.
- horizontal_flip is for randomly flipping half of the images horizontally. This is relevant when there are no assumptions of horizontal assymmetry (e.g. real-world pictures).
- fill_mode is the strategy used for filling in newly created pixels, which can appear after a rotation or a width/height shift.

Augmentation: A technique to avoid overfitting

```
model = tf.keras.models.Sequential([
    tf.keras.layers.Conv2D(32, (3,3), activation='relu', input_shape=(150, 150, 3)),
    tf.keras.layers.MaxPooling2D(2, 2),
    tf.keras.layers.Conv2D(64, (3,3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2,2),
    tf.keras.layers.Conv2D(128, (3,3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2,2),
    tf.keras.layers.Conv2D(128, (3,3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2,2),
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(512, activation='relu'),
    tf.keras.layers.Dense(1, activation='sigmoid')
])

model.compile(loss='binary_crossentropy',
              optimizer=RMSprop(lr=1e-4),
              metrics=['accuracy'])

# This code has changed. Now instead of the ImageGenerator just rescaling
# the image, we also rotate and do other operations
# Updated to do image augmentation
train_datagen = ImageDataGenerator(
      rescale=1./255,
      rotation_range=40,
      width_shift_range=0.2,
      height_shift_range=0.2,
      shear_range=0.2,
      zoom_range=0.2,
      horizontal_flip=True,
      fill_mode='nearest')

test_datagen = ImageDataGenerator(rescale=1./255)
```



tf.keras.preprocessing.image.ImageDataGenerator

`zoom_range`: Float or [lower, upper]. Range for random zoom. If a float, `[lower, upper] = [1-zoom_range, 1+zoom_range]`.

`fill_mode`: One of {"constant", "nearest", "reflect" or "wrap"}. Default is 'nearest'.Points outside the boundaries of the input are filled according to the given mode:

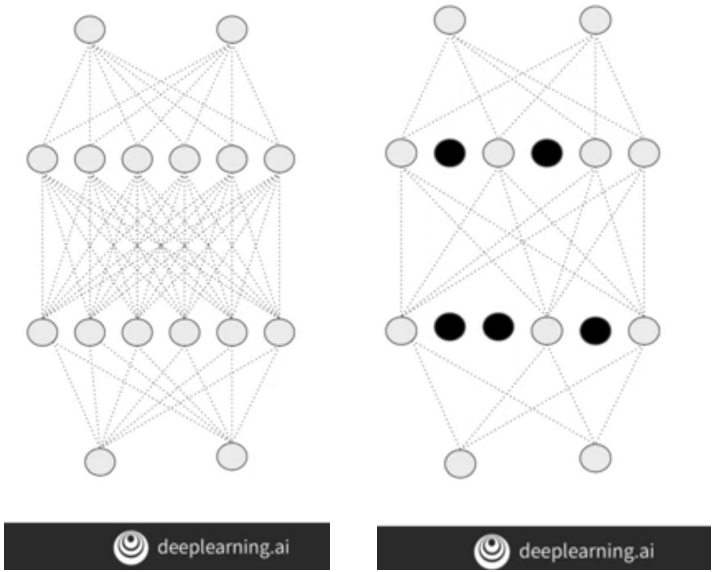
- 'constant': kkkkkkkk|abcd|kkkkkkkk (cval=k)
- 'nearest': aaaaaaaa|abcd|dddddddd
- 'reflect': abcddcba|abcd|dcbaabcd
- 'wrap': abcdabcd|abcd|abcdabcd

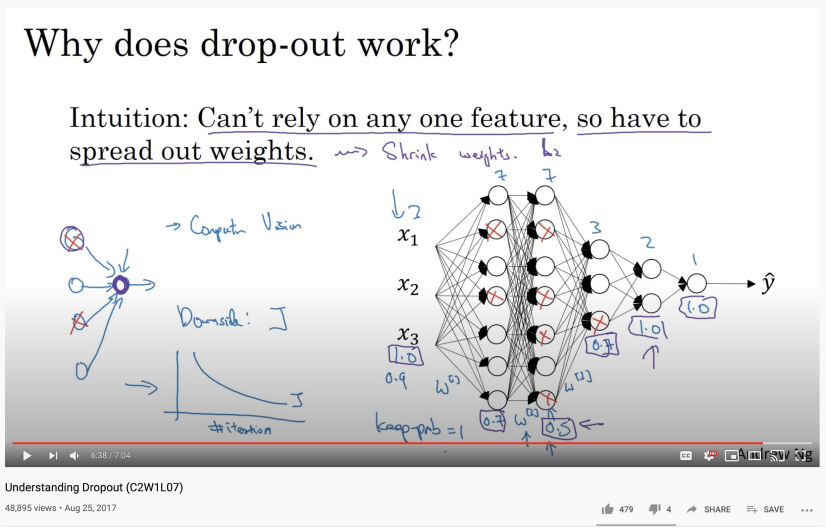
`cval`: Float or Int. Value used for points outside the boundaries when `fill_mode = "constant"`.

`preprocessing_function`: function that will be applied on each input. The function will run after the image is resized and augmented.

original	constant (cval=0)	constant (cval=100)
		
nearest	**reflect**	**wrap**
		

Augmentation: A technique to avoid overfitting





Source: https://www.youtube.com/watch?v=ARq74QuavAo

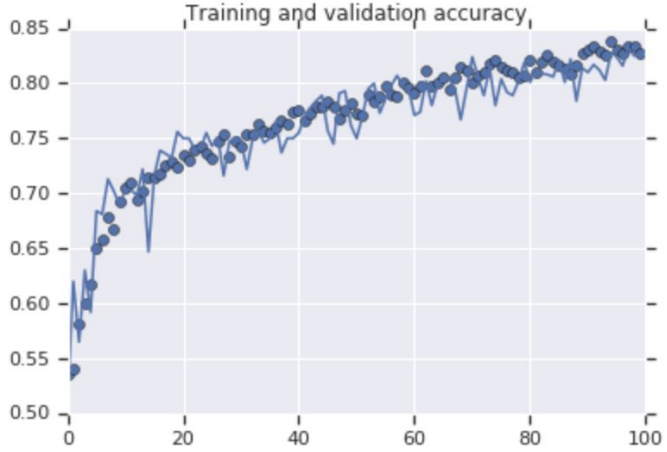
Another useful tool to explore at this point is the Dropout.

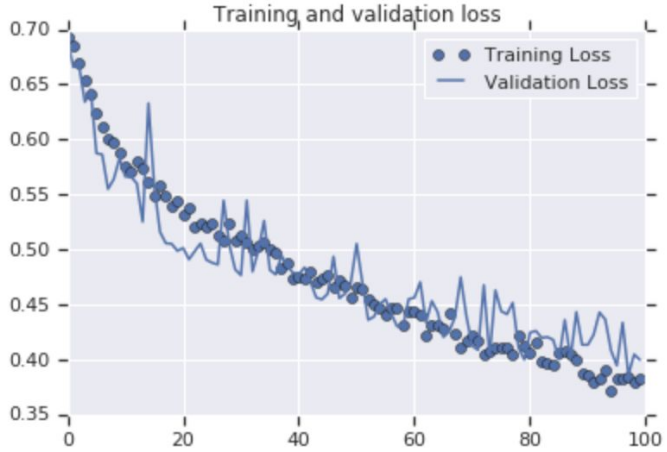
The idea behind Dropouts is that they remove a random number of neurons in your neural network. This works very well for two reasons: The first is that neighboring neurons often end up with similar weights, which can lead to overfitting, so dropping some out at random can remove this. The second is that often a neuron can over-weigh the input from a neuron in the previous layer, and can over specialize as a result. Thus, dropping out can break the neural network out of this potential bad habit!

Check out Andrew's terrific video explaining dropouts here: https://www.youtube.com/watch?v=ARq74QuavAo

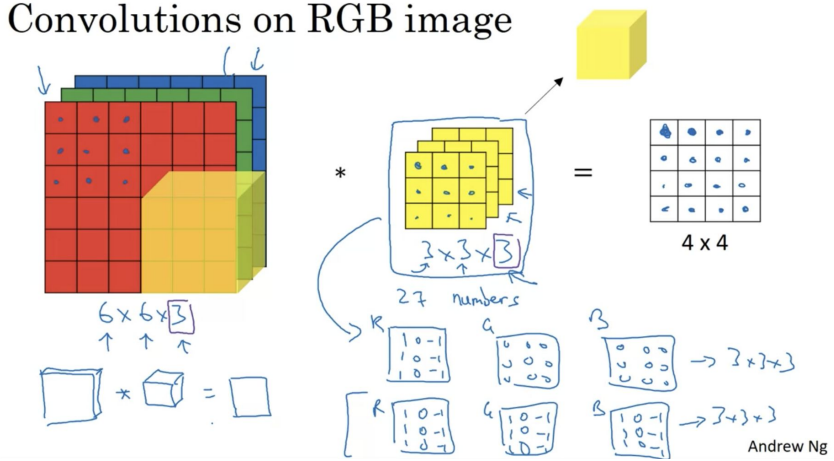
Augmentation: A technique to avoid overfitting

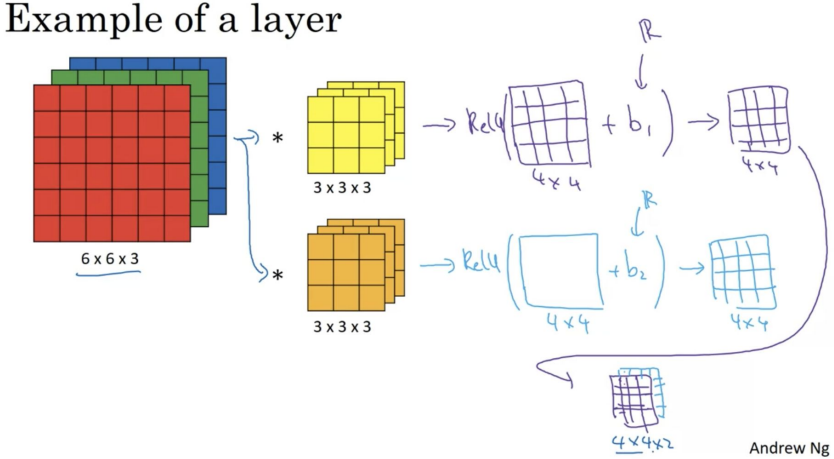
```
model = tf.keras.models.Sequential([
    tf.keras.layers.Conv2D(32, (3,3), activation='relu', input_shape=(150, 150, 3)),
    tf.keras.layers.MaxPooling2D(2, 2),
    tf.keras.layers.Conv2D(64, (3,3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2,2),
    tf.keras.layers.Conv2D(128, (3,3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2,2),
    tf.keras.layers.Conv2D(128, (3,3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2,2),
    tf.keras.layers.Dropout(0.5), #NEW here
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(512, activation='relu'),
    tf.keras.layers.Dense(1, activation='sigmoid')
])

model.compile(loss='binary_crossentropy',
              optimizer=RMSprop(lr=1e-4),
              metrics=['accuracy'])

# This code has changed. Now instead of the ImageGenerator just rescaling
# the image, we also rotate and do other operations
# Updated to do image augmentation
train_datagen = ImageDataGenerator(
      rescale=1./255,
      rotation_range=40,
      width_shift_range=0.2,
      height_shift_range=0.2,
      shear_range=0.2,
      zoom_range=0.2,
      horizontal_flip=True,
      fill_mode='nearest')

test_datagen = ImageDataGenerator(rescale=1./255)
```

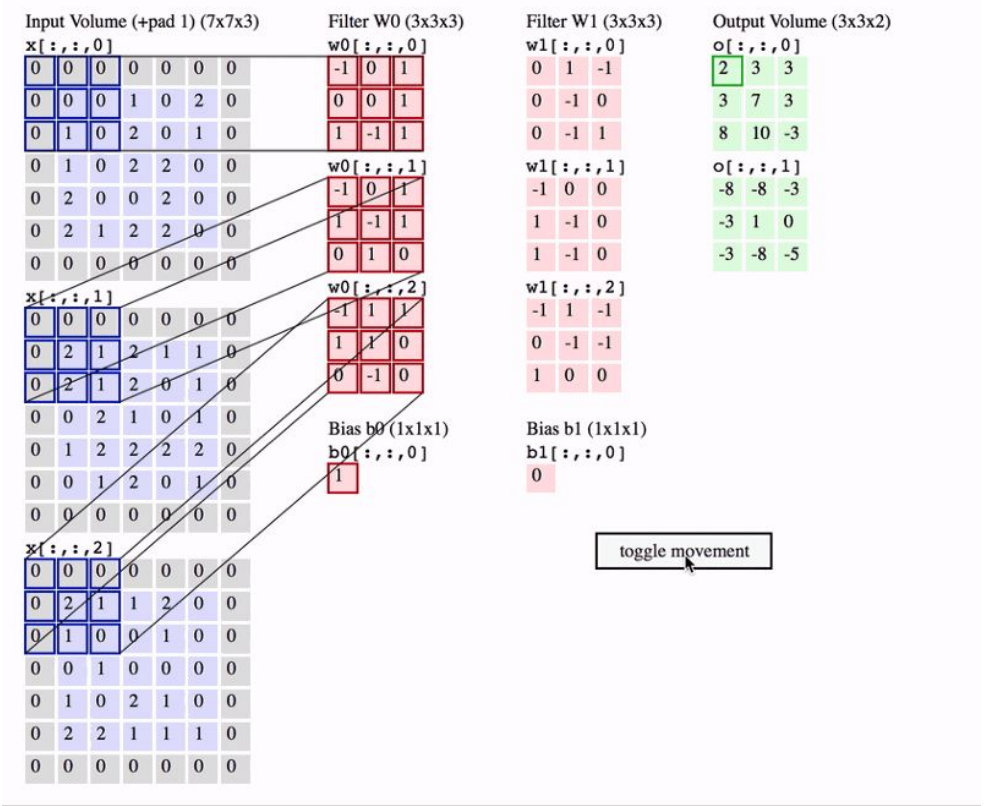




Content added after Week 1 and 2, based on group feedback





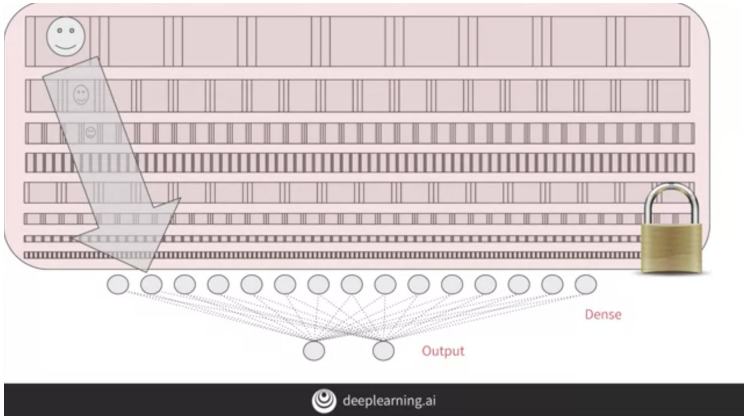


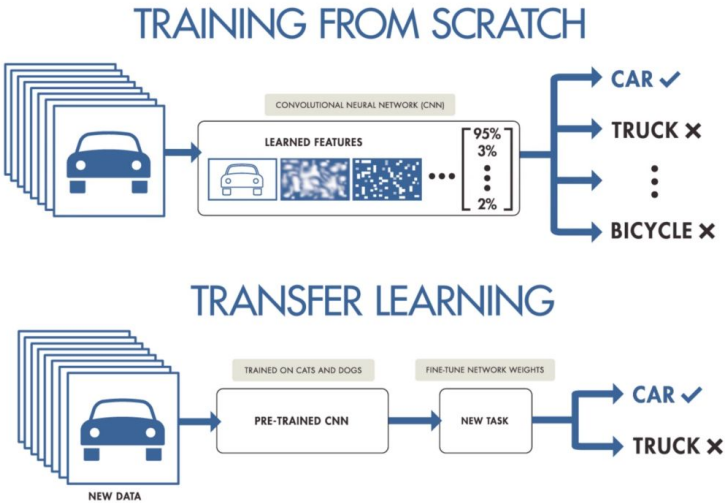
Source: https://www.coursera.org/learn/convolutional-neural-networks
https://images.app.goo.gl/Z4xDvjzi3Gcerpm27

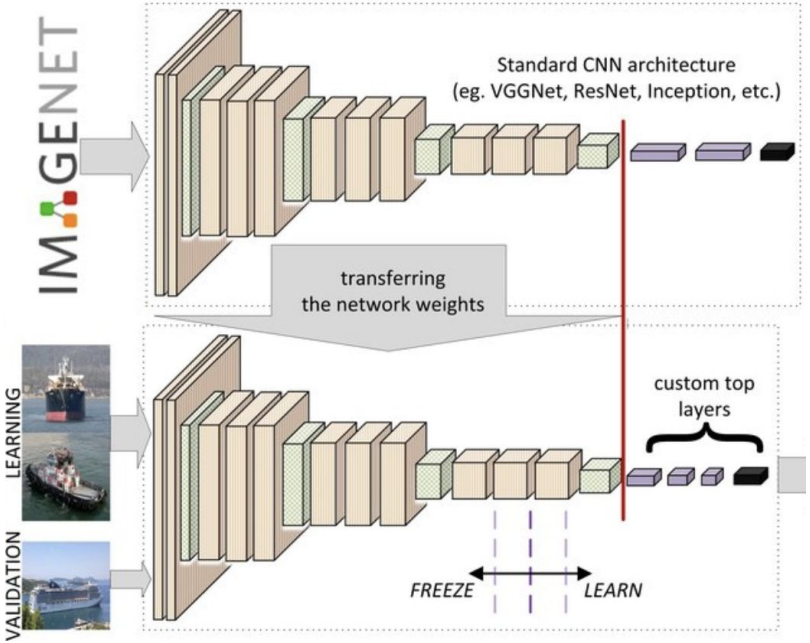
Course 2: Convolutional Neural Networks in TensorFlow

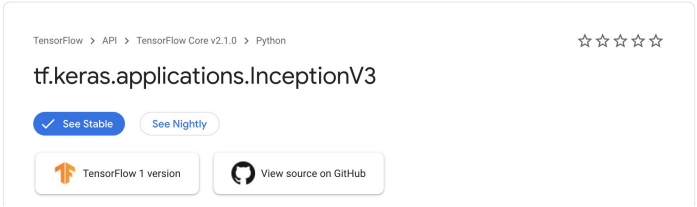
Week 1: Exploring a Larger Dataset
Week 2: Augmentation: A technique to avoid overfitting
Week 3: Transfer Learning
Week 4: Multiclass Classifications

Transfer Learning









Source:

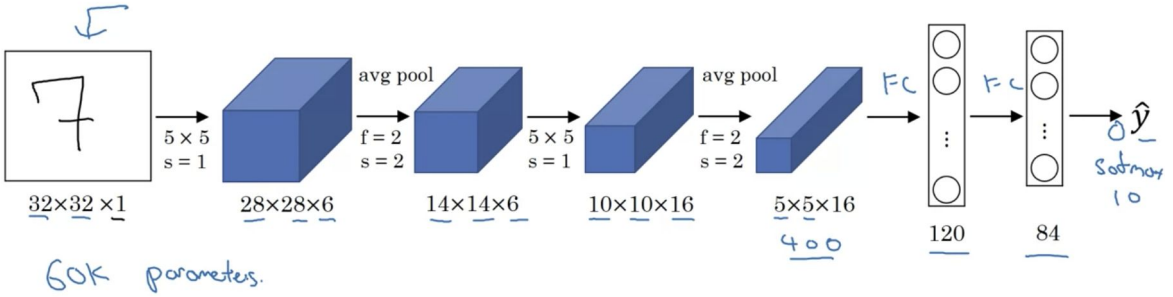
https://www.tensorflow.org/api_docs/python/tf/keras/applications/InceptionV3
https://mc.ai/introducing-transfer-learning-as-your-next-engine-to-drive-future-innovations-2/
https://www.researchgate.net/figure/Ensemble-transfer-learning-using-pretrained-CNN-model-initialized-with-weights-trained-on_fig4_333619654

Extra Content - Classic Networks

Gradient-Based Learning Applied to Document Recognition

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner

LeNet - 5 1998



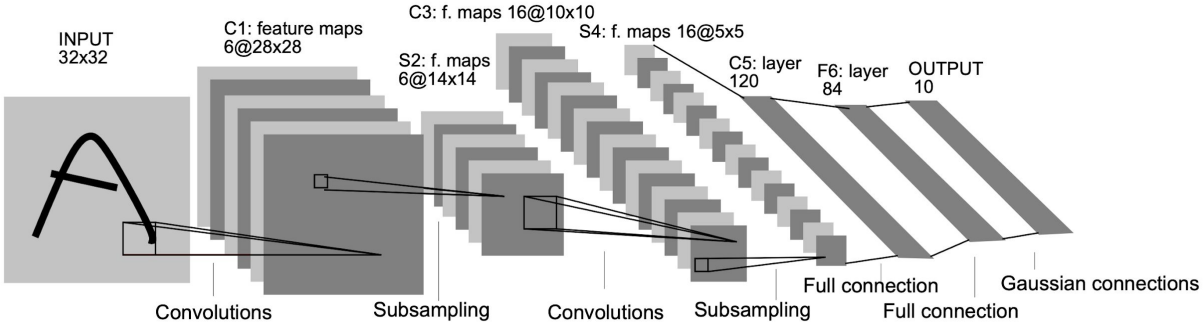


Fig. 2. Architecture of LeNet-5, a Convolutional Neural Network, here for digits recognition. Each plane is a feature map, i.e. a set of units whose weights are constrained to be identical.

Source:

https://www.coursera.org/learn/convolutional-neural-networks/lecture/MmYe2/classic-networks

http://yann.lecun.com/exdb/publis/pdf/lecun-01a.pdf

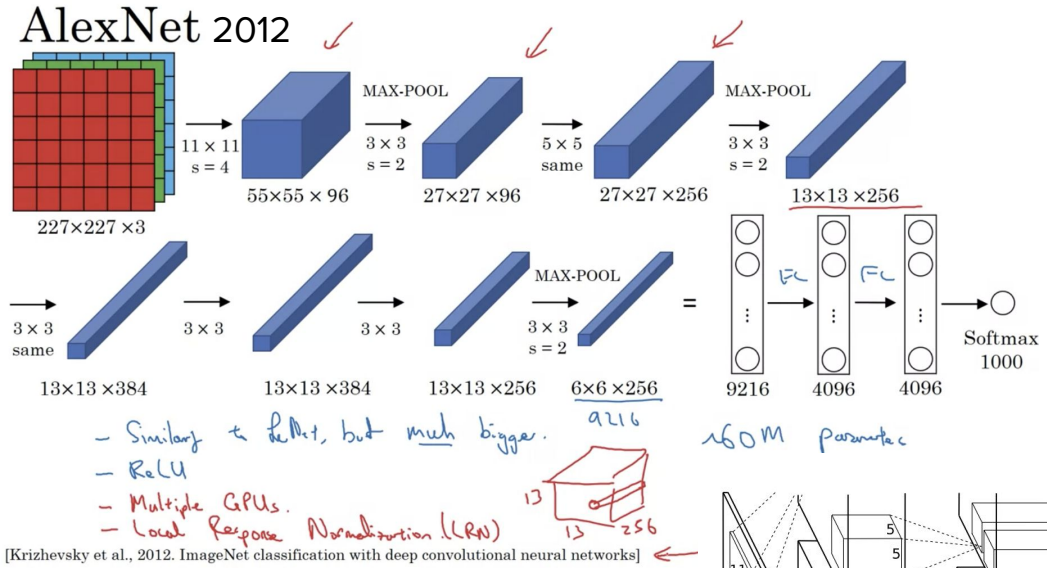
Extra Content - Classic Networks

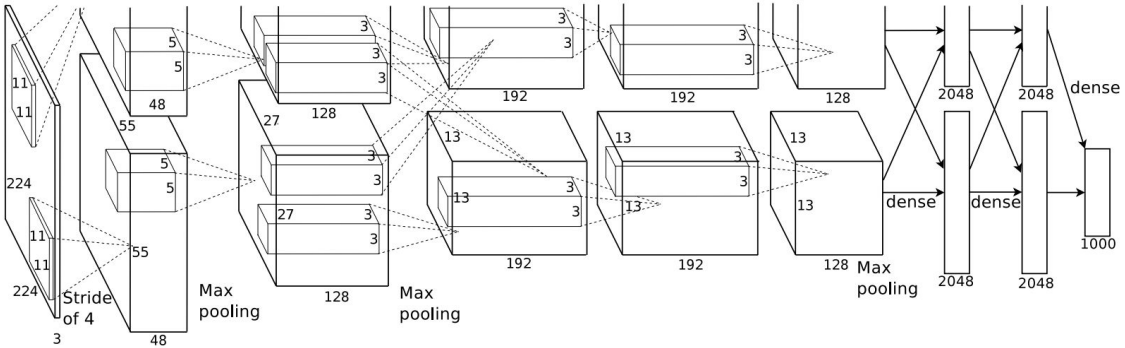
ImageNet Classification with Deep Convolutional Neural Networks

Alex Krizhevsky University of Toronto kriz@cs.utoronto.ca

Ilya Sutskever University of Toronto ilya@cs.utoronto.ca

Geoffrey E. Hinton University of Toronto hinton@cs.utoronto.ca





Source:

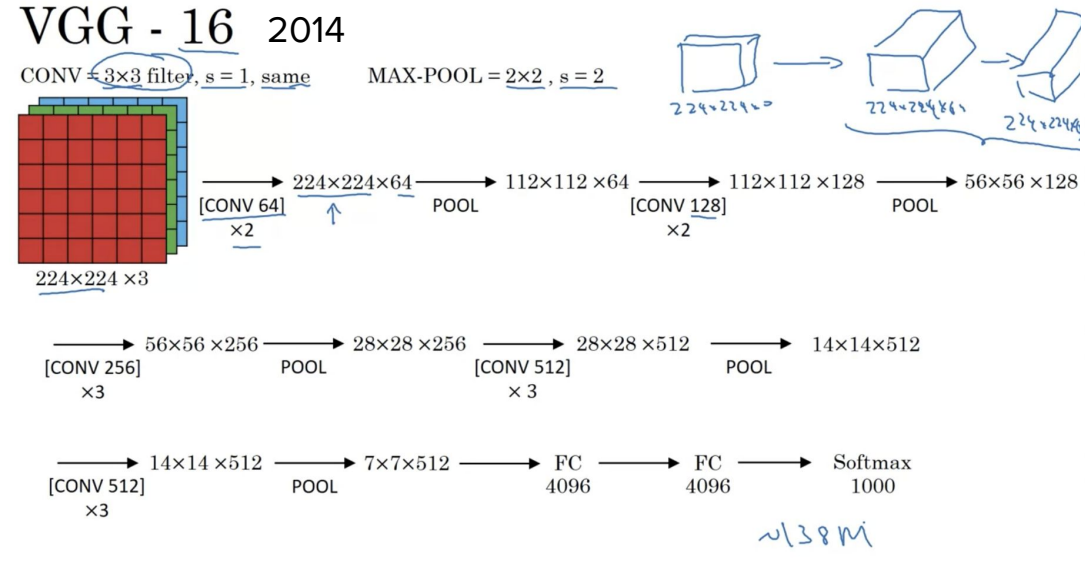
https://www.coursera.org/learn/convolutional-neural-networks/lecture/MmYe2/classic-networks

https://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf

Extra Content - Classic Networks

VERY DEEP CONVOLUTIONAL NETWORKS FOR LARGE-SCALE IMAGE RECOGNITION

Karen Simonyan* **& Andrew Zisserman+**
Visual Geometry Group, Department of Engineering Science, University of Oxford
{karen,az}@robots.ox.ac.uk



[Simonyan & Zisserman 2015. Very deep convolutional networks for large-scale image recognition]

Andrew Ng

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

Table 2: **Number of parameters** (in millions).

Network	A,A-LRN	B	C	D	E
Number of parameters	133	133	134	138	144

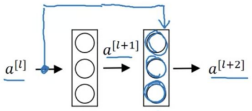
Source:
https://www.coursera.org/learn/convolutional-neural-networks/lecture/MmYe2/classic-networks
https://arxiv.org/pdf/1409.1556.pdf

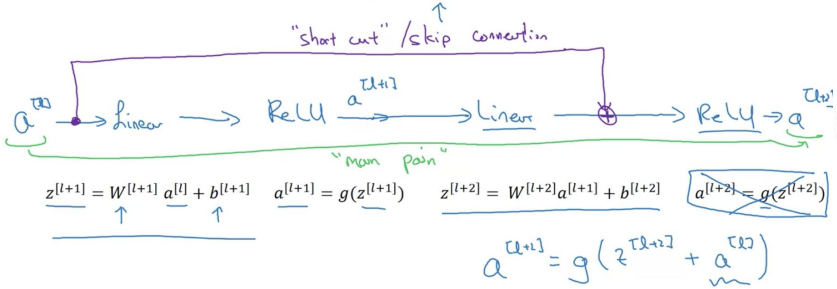
Extra Content - Classic Networks

Deep Residual Learning for Image Recognition 2015

Kaiming He Xiangyu Zhang Shaoqing Ren Jian Sun
Microsoft Research
{kahe, v-xiangz, v-shren, jiansun}@microsoft.com

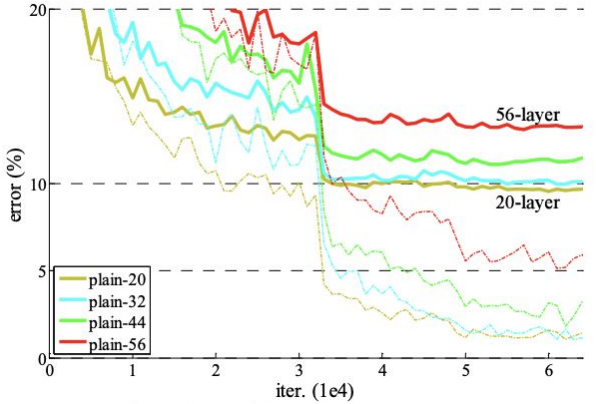
Residual block

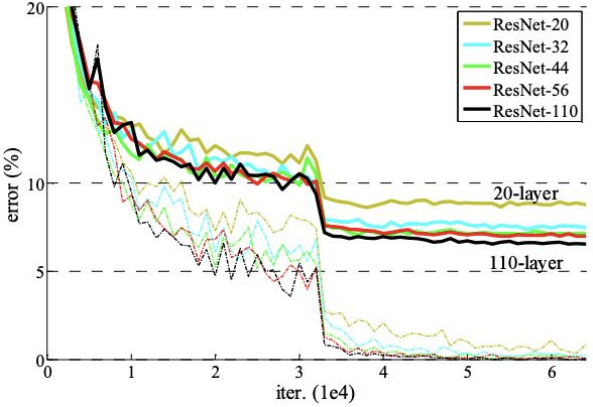


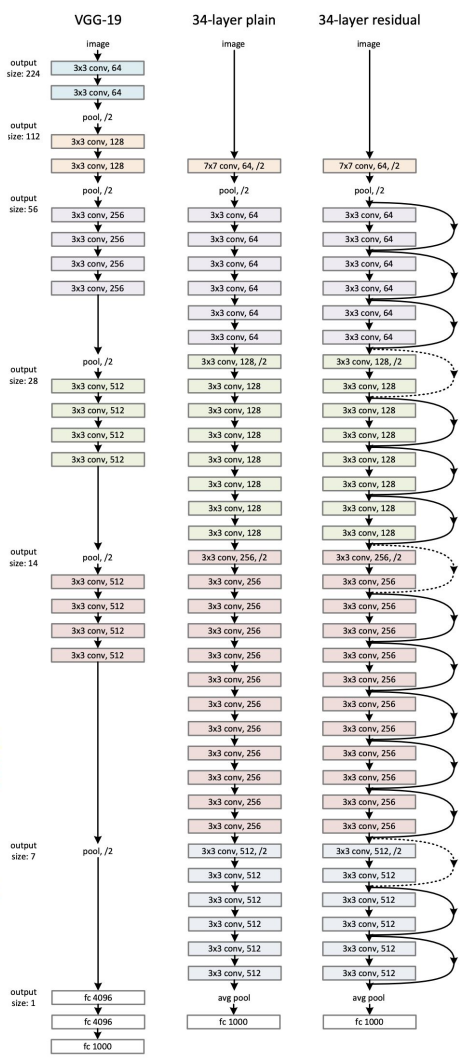


[He et al., 2015. Deep residual networks for image recognition]

Andrew Ng





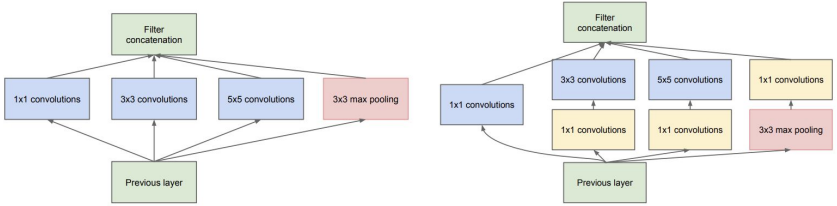


Source:
https://www.coursera.org/learn/convolutional-neural-networks/lecture/MmYe2/classic-networks
https://arxiv.org/pdf/1512.03385.pdf

Extra Content - Classic Networks

Going deeper with convolutions 2014

Christian Szegedy Google Inc.
Wei Liu University of North Carolina, Chapel Hill
Yangqing Jia Google Inc.
Pierre Sermanet Google Inc.
Scott Reed University of Michigan
Dragomir Anguelov Google Inc.
Dumitru Erhan Google Inc.
Vincent Vanhoucke Google Inc.
Andrew Rabinovich Google Inc.



(a) Inception module, naïve version (b) Inception module with dimension reductions

Figure 2: Inception module

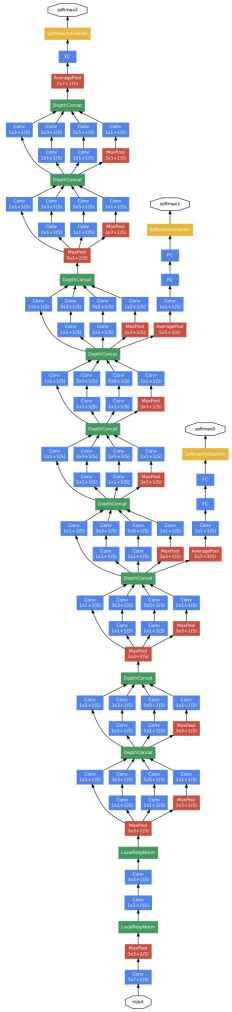
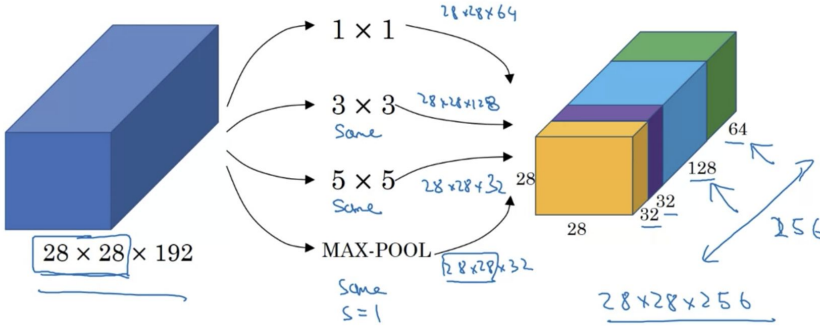
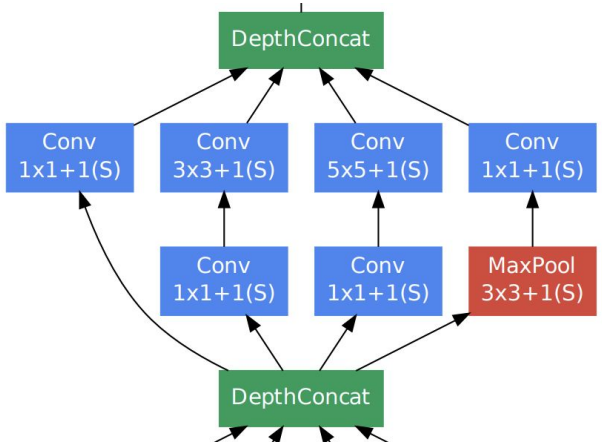


Figure 3: GoogLeNet network with all the bells and whistles

Motivation for inception network



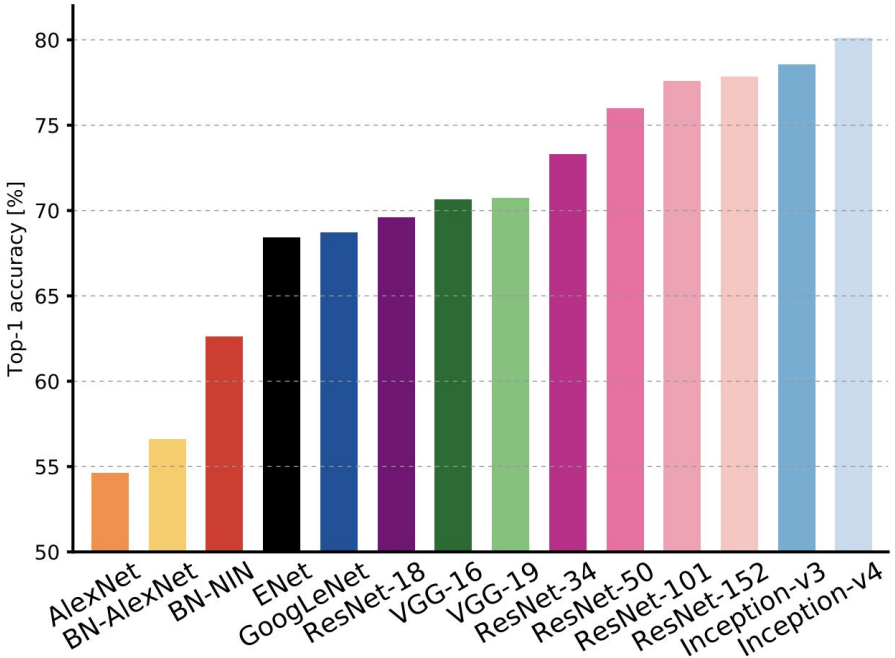
[Szegedy et al. 2014. Going deeper with convolutions]

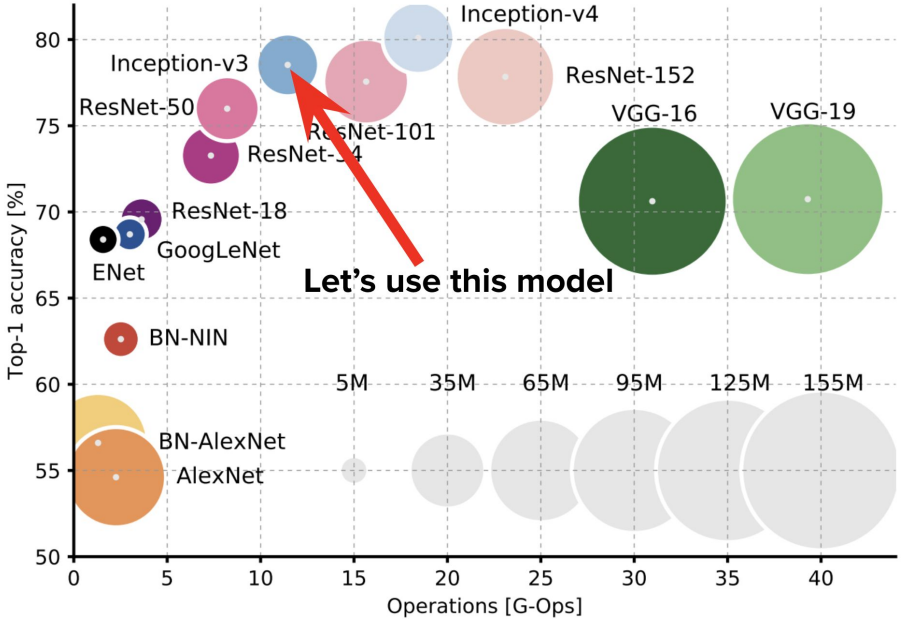


Andrew Ng

Source:
https://www.coursera.org/learn/convolutional-neural-networks/lecture/5WIZm/inception-network-motivation
https://arxiv.org/pdf/1409.4842.pdf

Extra Content - Classic Networks





Source: https://arxiv.org/pdf/1605.07678.pdf

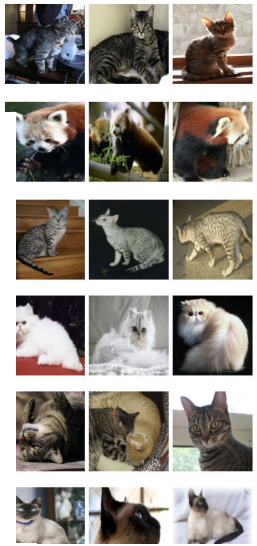
Transfer Learning

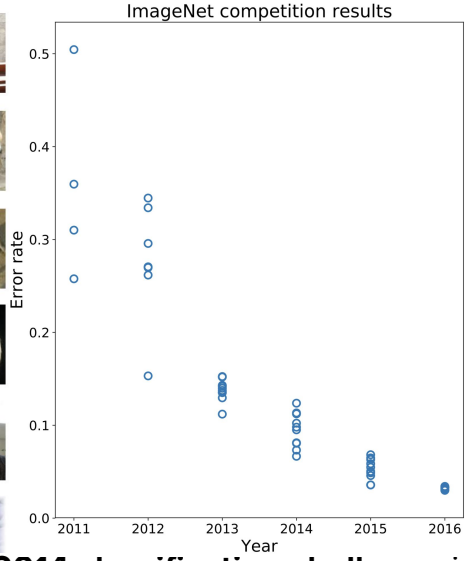
Start exploring here

Numbers in brackets: (the number of synsets in the subtree).

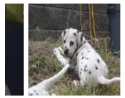
- ImageNet 2011 Fall Release (32326)
  - plant, flora, plant life (4486)
  - geological formation, formation (175)
  - natural object (1112)
  - sport, athletics (176)
  - artifact, artefact (10504)
  - fungus (308)
  - person, individual, someone, somebody, mortal, soul (6978)
  - animal, animate being, beast, brute, creature, fauna (3998)
  - Misc (20400)

Source: http://www.image-net.org





Synset: Newfoundland, Newfoundland dog
Definition: a breed of very large heavy dogs with a vigorous swimmers; developed in Newfoundland.
Popularity percentile:: 70%
Depth in WordNet: 8



Synset: dalmatian, coach dog, carriage dog
Definition: a large breed having a smooth white coa
Popularity percentile:: 65%
Depth in WordNet: 8

Synset: Eskimo dog, husky
Definition: breed of heavy-coated Arctic sled dog.
Popularity percentile:: 62%
Depth in WordNet: 9

Synset: German shepherd, German shepherd dog,
Definition: breed of large shepherd dogs used in po
Popularity percentile:: 61%
Depth in WordNet: 10

Synset: African hunting dog, hyena dog, Cape hunt
Definition: a powerful doglike mammal of southern area.
Popularity percentile:: 58%

The **ILSVRC 2014 classification challenge** involves the task of classifying the image into one of **1000** leaf-node categories in the **Imagenet** hierarchy. There are about **1.2 million images for training**, **50,000 for validation** and **100,000 images for testing**. Each image is associated with one ground truth category, and performance is measured based on the highest scoring classifier predictions.

top-1 accuracy rate compares the ground truth against the **first** predicted class

top-5 error rate compares the ground truth against the **first 5** predicted classes. An image is deemed correctly classified if the ground truth is among the top-5, regardless of its rank in them. The challenge uses the top-5 error rate for ranking purposes.

Transfer Learning

```
from tensorflow.keras import layers
from tensorflow.keras import Model
!wget --no-check-certificate \
    https://storage.googleapis.com/mledu-datasets/inception_v3_weights_tf_dim_ordering_tf_kernels_notop.h5 \
    -O /tmp/inception_v3_weights_tf_dim_ordering_tf_kernels_notop.h5

from tensorflow.keras.applications.inception_v3 import InceptionV3

local_weights_file = '/tmp/inception_v3_weights_tf_dim_ordering_tf_kernels_notop.h5'
```

```
pre_trained_model = InceptionV3(input_shape = (150, 150, 3),
                                include_top = False,
                                weights = None)

pre_trained_model.load_weights(local_weights_file)

for layer in pre_trained_model.layers:
  layer.trainable = False

#pre_trained_model.summary()

last_layer = pre_trained_model.get_layer('mixed7')
print('last layer output shape: ', last_layer.output_shape)
last_output = last_layer.output
```

```
last layer output shape:

(None, 7, 7, 768)
```

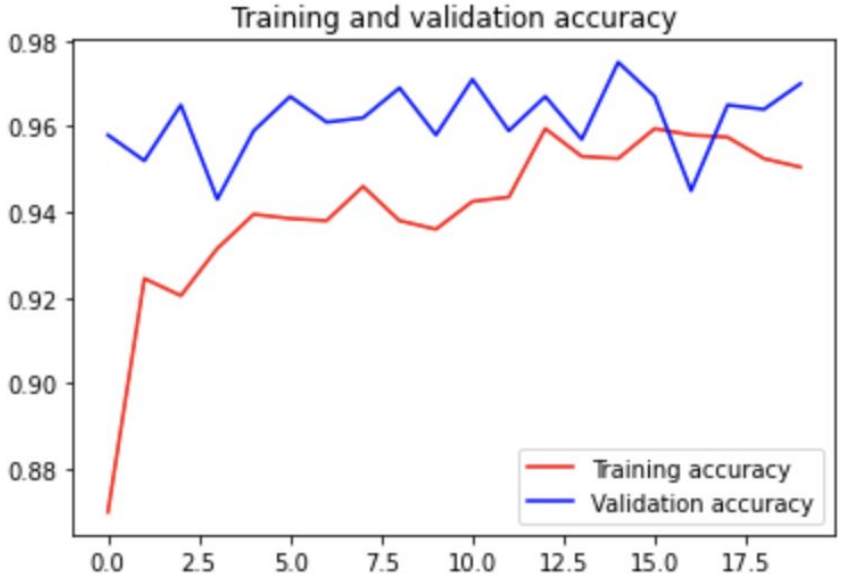
```
from tensorflow.keras.optimizers import RMSprop

# Flatten the output layer to 1 dimension
x = layers.Flatten()(last_output)
# Add a fully connected layer with 1,024
# hidden units and ReLU activation
x = layers.Dense(1024, activation='relu')(x)
# Add a dropout rate of 0.2
x = layers.Dropout(0.2)(x)
# Add a final sigmoid layer for classification
x = layers.Dense  (1, activation='sigmoid')(x)

model = Model( pre_trained_model.input, x)

model.compile(optimizer = RMSprop(lr=0.0001),
              loss = 'binary_crossentropy',
              metrics = ['accuracy'])
```

```
pre_trained_model.summary()
```

mixed7 (Concatenate)	(None, 7, 7, 768)	0	activation_60[0][0] activation_63[0][0] activation_68[0][0] activation_69[0][0]
conv2d_72 (Conv2D)	(None, 7, 7, 192)	147456	mixed7[0][0]
batch_normalization_72 (BatchNo	(None, 7, 7, 192)	576	conv2d_72[0][0]
activation_72 (Activation)	(None, 7, 7, 192)	0	batch_normalization_72[0][0]
conv2d_73 (Conv2D)	(None, 7, 7, 192)	258048	activation_72[0][0]
batch_normalization_73 (BatchNo	(None, 7, 7, 192)	576	conv2d_73[0][0]
activation_73 (Activation)	(None, 7, 7, 192)	0	batch_normalization_73[0][0]
conv2d_70 (Conv2D)	(None, 7, 7, 192)	147456	mixed7[0][0]
conv2d_74 (Conv2D)	(None, 7, 7, 192)	258048	activation_73[0][0]
batch_normalization_70 (BatchNo	(None, 7, 7, 192)	576	conv2d_70[0][0]
batch_normalization_74 (BatchNo	(None, 7, 7, 192)	576	conv2d_74[0][0]
activation_70 (Activation)	(None, 7, 7, 192)	0	batch_normalization_70[0][0]
activation_74 (Activation)	(None, 7, 7, 192)	0	batch_normalization_74[0][0]
conv2d_71 (Conv2D)	(None, 3, 3, 320)	552960	activation_70[0][0]
conv2d_75 (Conv2D)	(None, 3, 3, 192)	331776	activation_74[0][0]
batch_normalization_71 (BatchNo	(None, 3, 3, 320)	960	conv2d_71[0][0]
batch_normalization_75 (BatchNo	(None, 3, 3, 192)	576	conv2d_75[0][0]
activation_71 (Activation)	(None, 3, 3, 320)	0	batch_normalization_71[0][0]
activation_75 (Activation)	(None, 3, 3, 192)	0	batch_normalization_75[0][0]
max_pooling2d_3 (MaxPooling2D)	(None, 3, 3, 768)	0	mixed7[0][0]
mixed8 (Concatenate)	(None, 3, 3, 1280)	0	activation_71[0][0] activation_75[0][0] max_pooling2d_3[0][0]

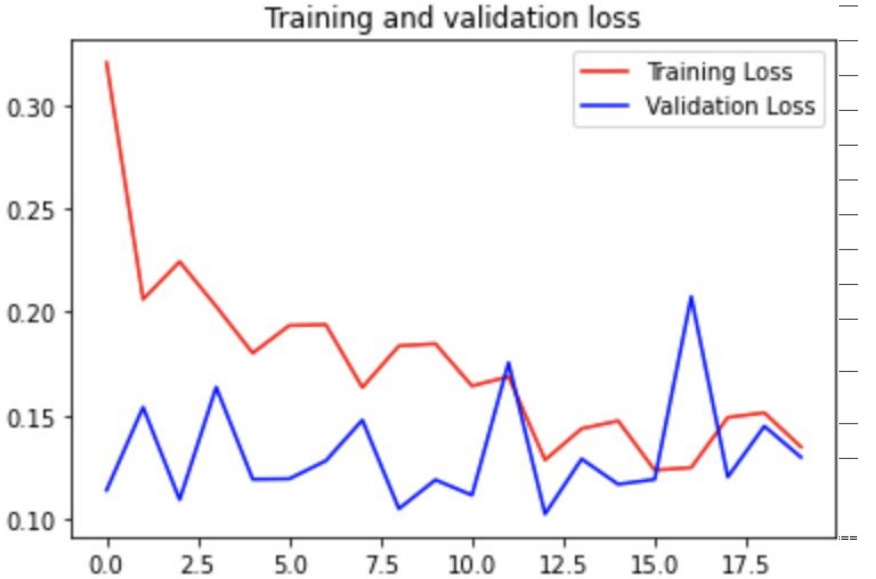
Transfer Learning

```
history = model.fit(
            train_generator,
            validation_data = validation_generator,
            steps_per_epoch = 100,
            epochs = 20,
            validation_steps = 50,
            verbose = 2)
```




```
pre_trained_model.summary()
```

activation_86 (Activation)	(None, 3, 3, 384)	0	batch_normalization_86[0][0]
activation_90 (Activation)	(None, 3, 3, 384)	0	batch_normalization_90[0][0]
conv2d_87 (Conv2D)	(None, 3, 3, 384)	442368	activation_86[0][0]
conv2d_88 (Conv2D)	(None, 3, 3, 384)	442368	activation_86[0][0]
conv2d_91 (Conv2D)	(None, 3, 3, 384)	442368	activation_90[0][0]
conv2d_92 (Conv2D)	(None, 3, 3, 384)	442368	activation_90[0][0]
average_pooling2d_8 (AveragePoo	(None, 3, 3, 2048)	0	mixed9[0][0]
conv2d_85 (Conv2D)	(None, 3, 3, 320)	655360	mixed9[0][0]
batch_normalization_87 (BatchNo	(None, 3, 3, 384)	1152	conv2d_87[0][0]
batch_normalization_88 (BatchNo	(None, 3, 3, 384)	1152	conv2d_88[0][0]
batch_normalization_91 (BatchNo	(None, 3, 3, 384)	1152	conv2d_91[0][0]

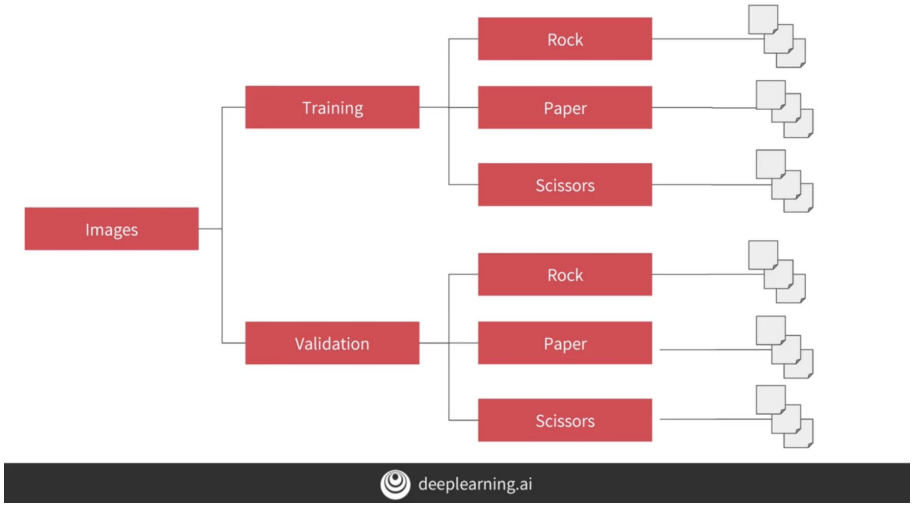


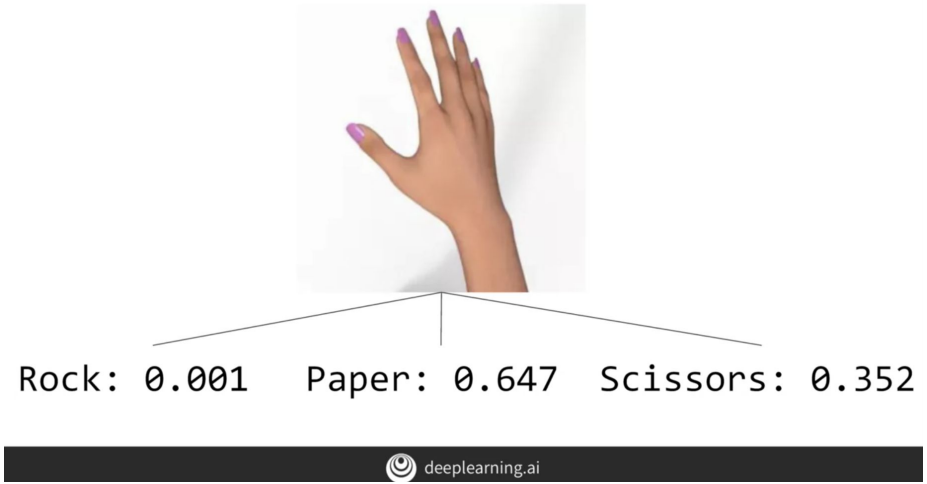
Non-trainable params: 21,802,784

Course 2: Convolutional Neural Networks in TensorFlow

Week 1: Exploring a Larger Dataset
Week 2: Augmentation: A technique to avoid overfitting
Week 3: Transfer Learning
Week 4: Multiclass Classifications

Multiclass Classifications





Rock Paper Scissors Dataset

Introducing Rock Paper Scissors – A multi class learning dataset

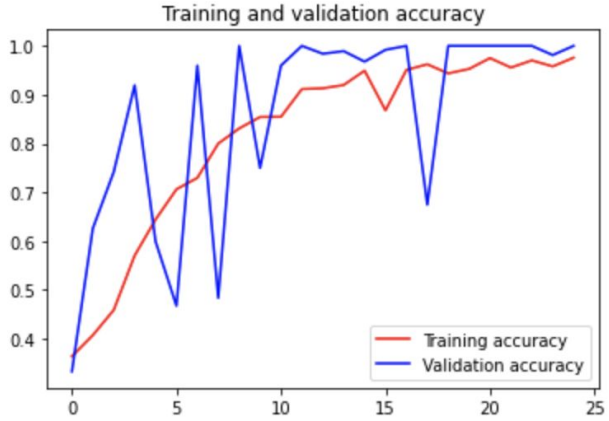
Abstract

Rock Paper Scissors is a dataset containing 2,892 images of diverse hands in Rock/Paper/Scissors poses.

Source: http://www.laurencemoroney.com/rock-paper-scissors-dataset

Multiclass Classifications

```
train_generator = training_datagen.flow_from_directory(
  TRAINING_DIR,
  batch_size = 20,
  target_size=(150,150),
  class_mode='categorical'
)

validation_generator = validation_datagen.flow_from_directory(
  VALIDATION_DIR,
  batch_size = 20,
  target_size=(150,150),
  class_mode='categorical'
)
```



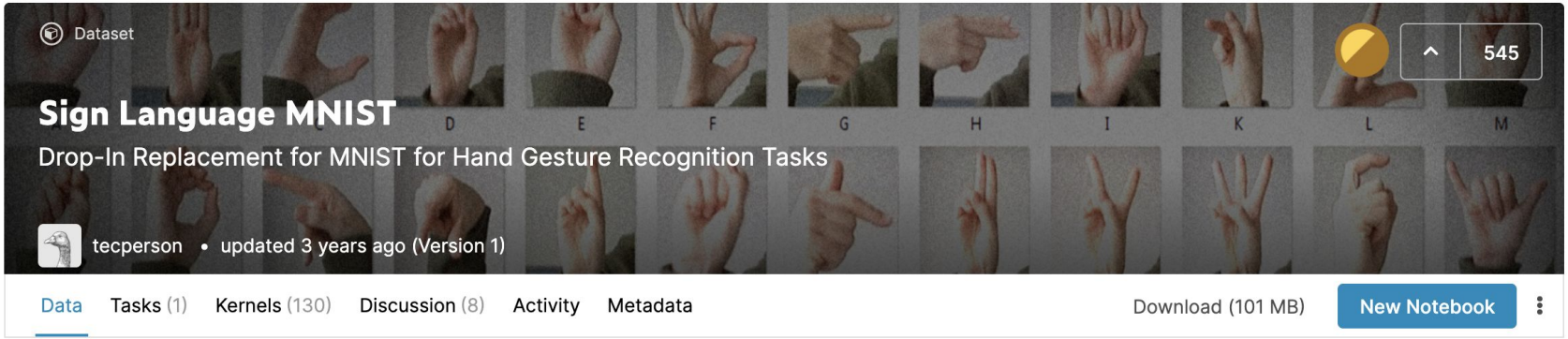

```
model = tf.keras.models.Sequential([
    # Note the input shape is the desired size of the image 150x150 with 3 bytes color
    # This is the first convolution
    tf.keras.layers.Conv2D(64, (3,3), activation='relu', input_shape=(150, 150, 3)),
    tf.keras.layers.MaxPooling2D(2, 2),
    # The second convolution
    tf.keras.layers.Conv2D(64, (3,3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2,2),
    # The third convolution
    tf.keras.layers.Conv2D(128, (3,3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2,2),
    # The fourth convolution
    tf.keras.layers.Conv2D(128, (3,3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2,2),
    # Flatten the results to feed into a DNN
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dropout(0.5),
    # 512 neuron hidden layer
    tf.keras.layers.Dense(512, activation='relu'),
    tf.keras.layers.Dense(3, activation='softmax')
])
```

Loss Functions for Multi-Class:

- One-Hot Encoding: categorical_crossentropy
- Integer Representation: sparse_categorical_crossentropy

```
model.compile(loss = 'categorical_crossentropy', optimizer='rmsprop', metrics=['accuracy'])
```

Multiclass Classifications

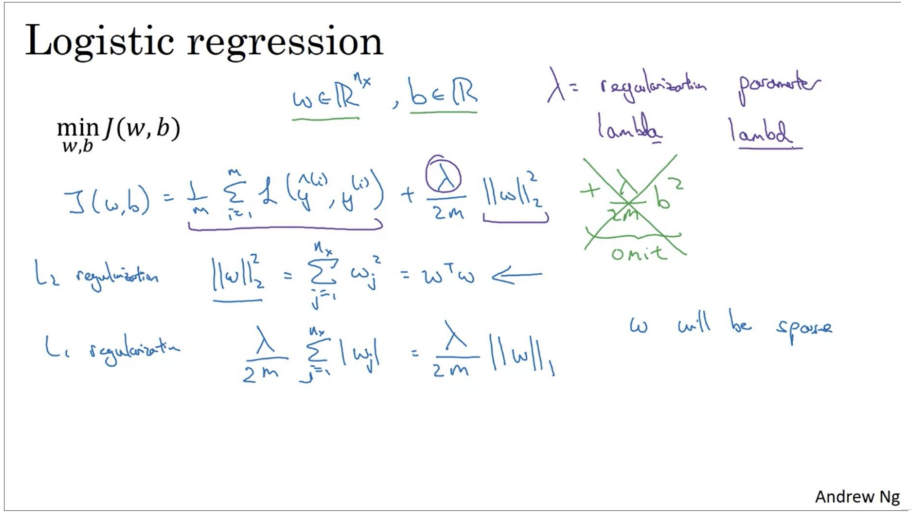


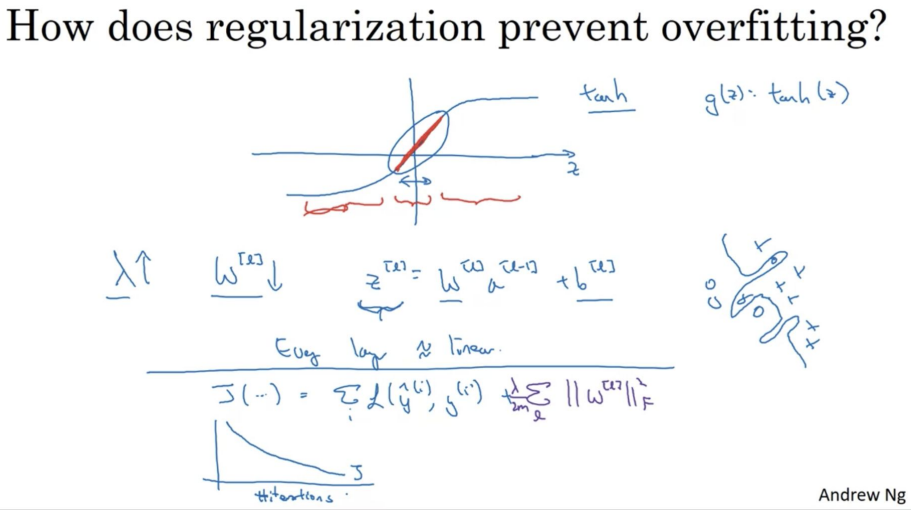
The American Sign Language letter database of hand gestures represent a multi-class problem with 24 classes of letters (excluding J and Z which require motion).
Source: https://www.kaggle.com/datamunge/sign-language-mnist

Course 2: Convolutional Neural Networks in TensorFlow

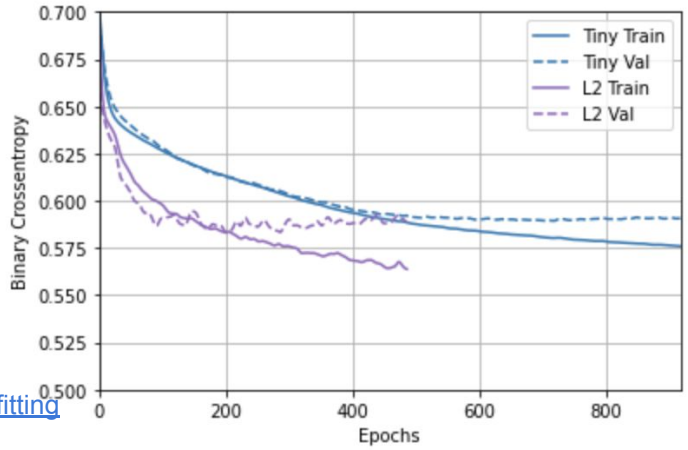
Week 1: Exploring a Larger Dataset
Week 2: Augmentation: A technique to avoid overfitting
Week 3: Transfer Learning
Week 4: Multiclass Classifications

Extra Content - Regularization



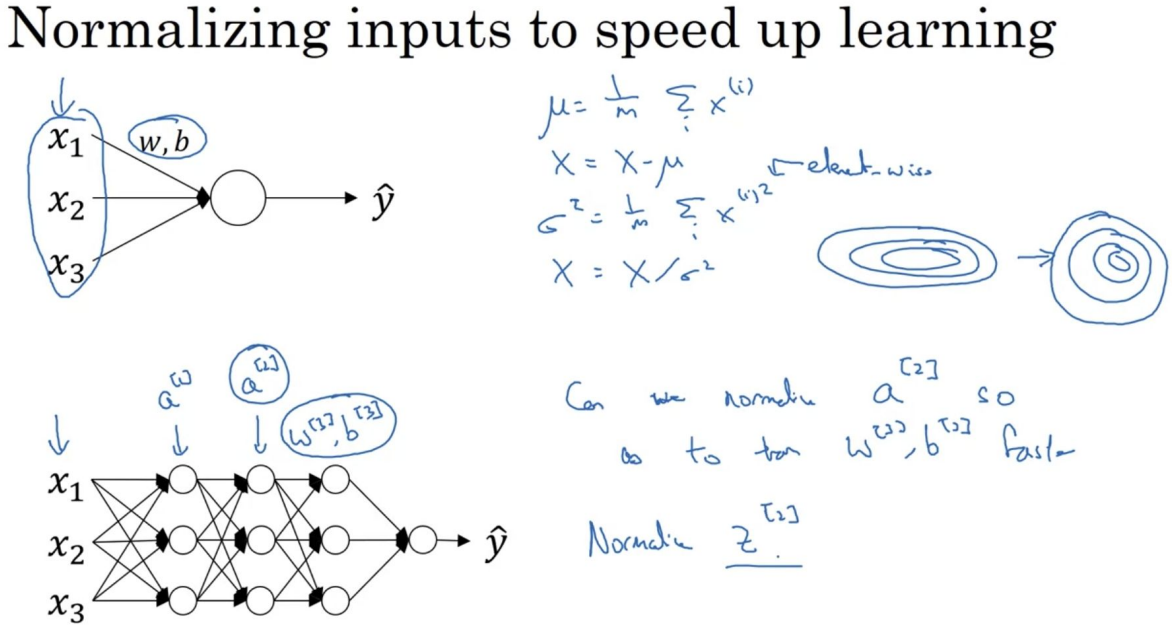



```
from tensorflow.keras import regularizers

l2_model = tf.keras.Sequential([
    layers.Dense(512, activation='elu',
                 kernel_regularizer=regularizers.l2(0.001),
                 input_shape=(FEATURES,)),
    layers.Dense(512, activation='elu',
                 kernel_regularizer=regularizers.l2(0.001)),
    layers.Dense(512, activation='elu',
                 kernel_regularizer=regularizers.l2(0.001)),
    layers.Dense(512, activation='elu',
                 kernel_regularizer=regularizers.l2(0.001)),
    layers.Dense(1)
])
```



Source: https://www.coursera.org/learn/deep-neural-network/lecture/Srsrc/regularization
https://www.coursera.org/learn/deep-neural-network/lecture/T6OJj/why-regularization-reduces-overfitting
https://www.tensorflow.org/tutorials/keras/overfit_and_underfit

Extra Content - Batch Normalization



TensorFlow > API > TensorFlow Core v2.1.0 > Python

tf.keras.layers.BatchNormalization

```
tf.keras.layers.BatchNormalization(
```

Normalize the activations of the previous layer at each batch, i.e. applies a transformation that maintains the mean activation close to 0 and the activation standard deviation close to 1.

Batch normalization differs from other layers in several key aspects:

Source:
https://www.coursera.org/learn/deep-neural-network/lecture/4ptp2/normalizing-activations-in-a-network
https://www.tensorflow.org/api_docs/python/tf/keras/layers/BatchNormalization

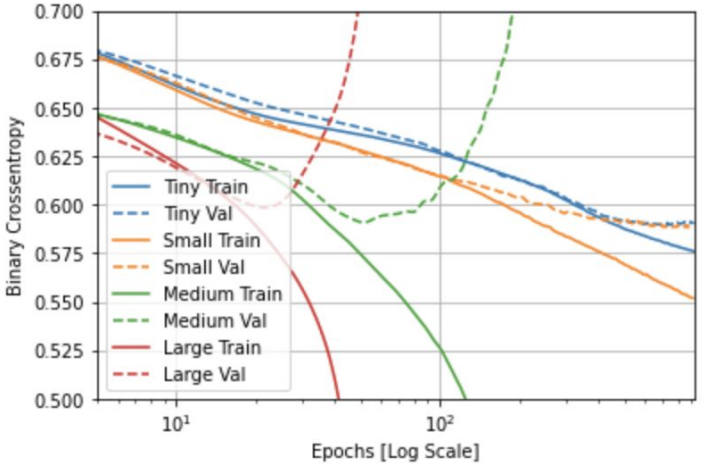
Extra Content - Overfitting in Neural Networks

To recap: here are the most common ways to prevent overfitting in neural networks:

- Get more training data.
- Reduce the capacity of the network.
- Add weight regularization.
- Add dropout.

The Higgs Dataset: 11M samples, 28 features, binary class

Total params:
Tiny: 0.5K
Small: 0.7K
Medium: 10K
Large: 800K

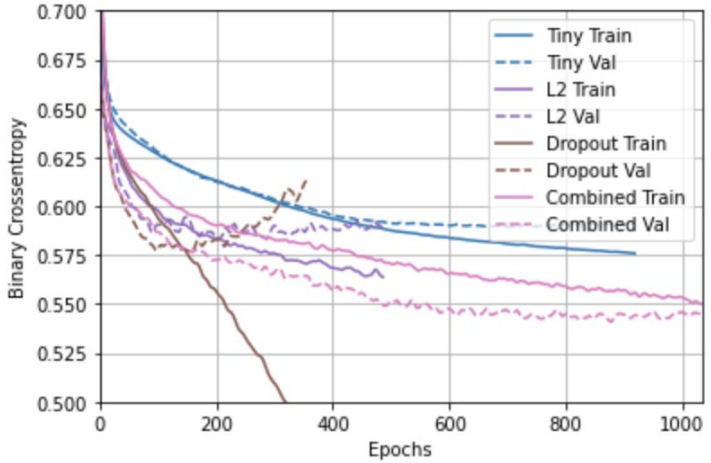


Two important approaches not covered in this guide are:

- data-augmentation
- batch normalization

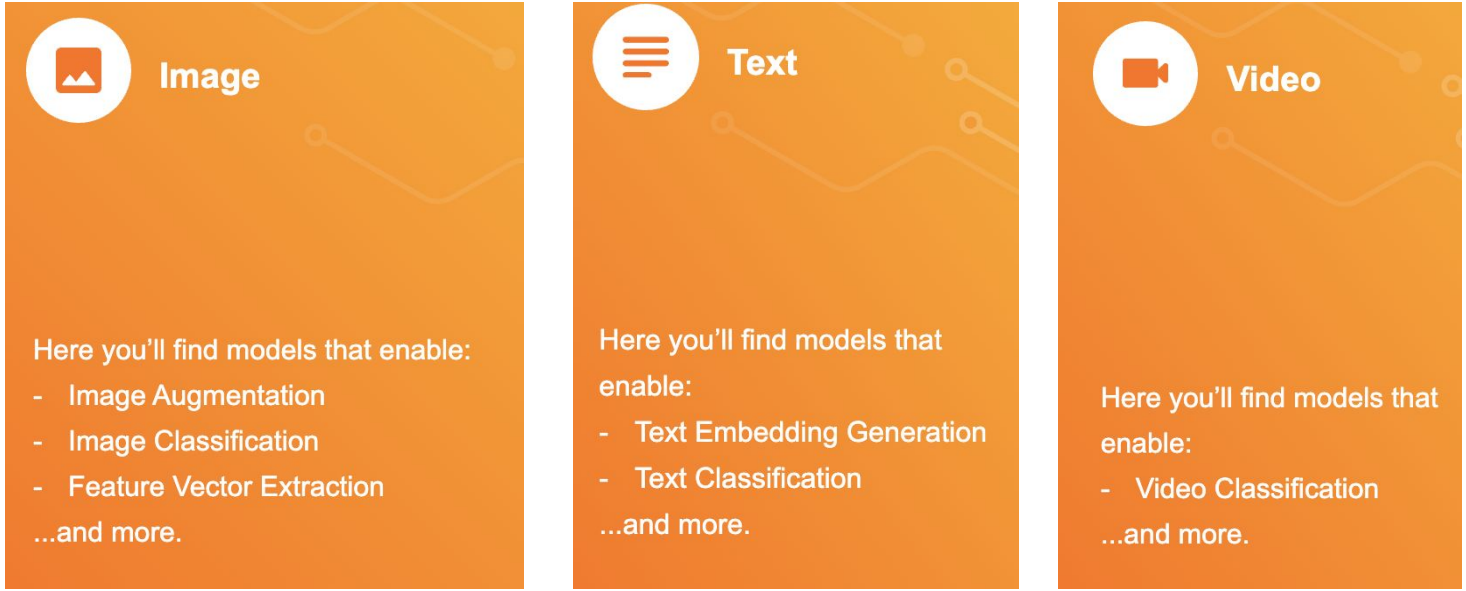
Remember that each method can help on its own, but often combining them can be even more effective.

- Data Augmentation
- Dropout
- Early Stopping
- Ensembling
- Injecting Noise
- L1 Regularization
- L2 Regularization



Source: https://www.tensorflow.org/tutorials/keras/overfit_and_underfit
https://ml-cheatsheet.readthedocs.io/en/latest/regularization.html

Content added after Week 3 and 4, based on group feedback

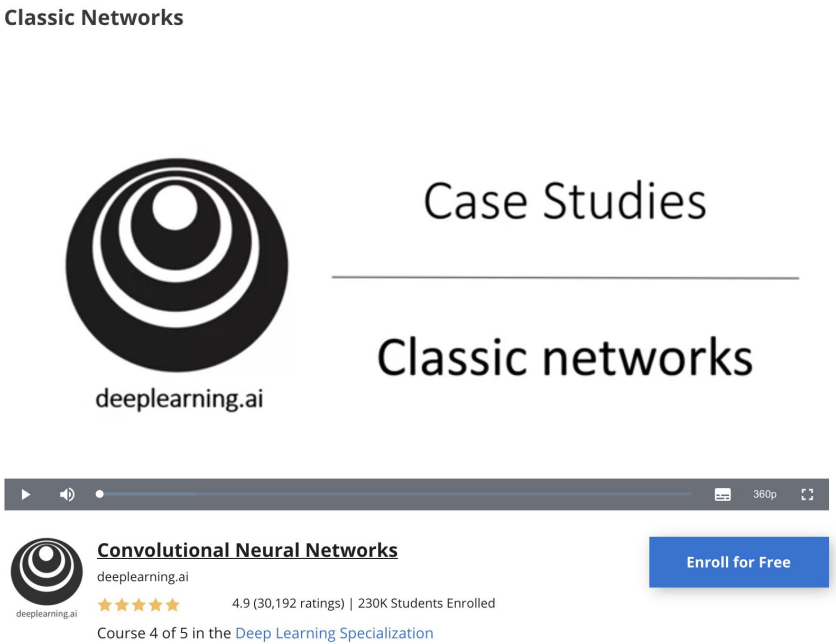


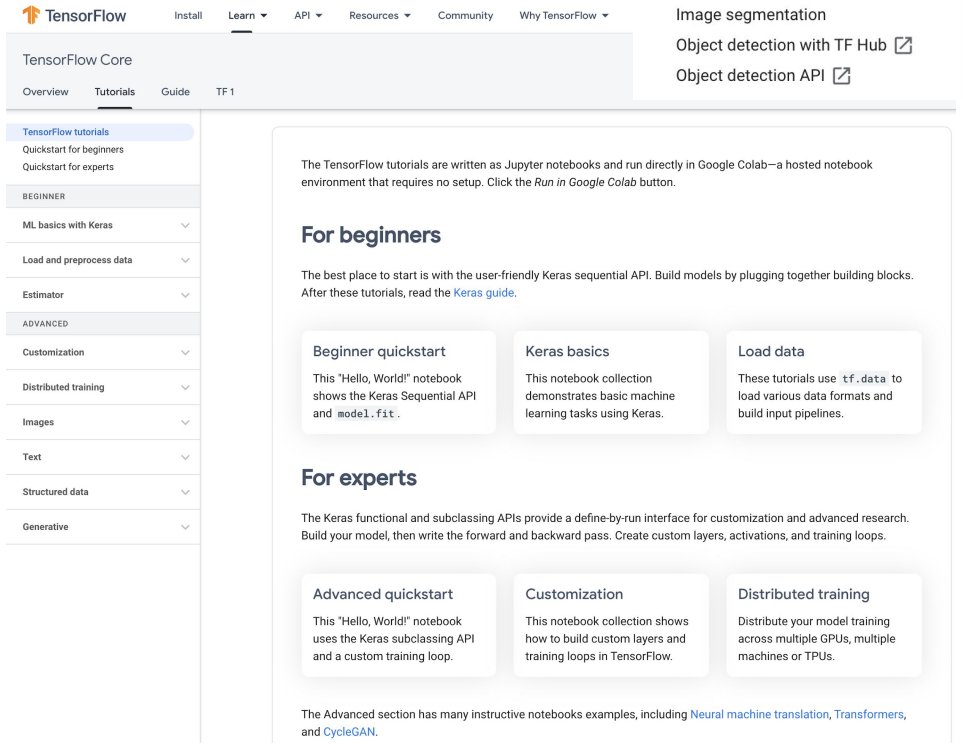
Source: https://tfhub.dev

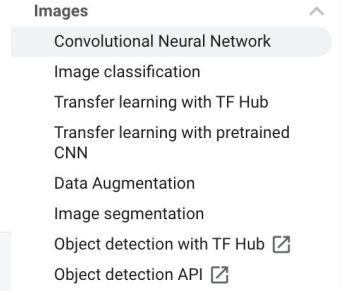
Course 2: Convolutional Neural Networks in TensorFlow - Questions

1. ...
2. ...
3. ...

Check out these resources







Source:
https://www.coursera.org/lecture/convolutional-neural-networks/classic-networks-MmYe2
https://www.tensorflow.org/tutorials/

Check out these events and Meetups

New AI For Medicine Specialization coming soon!

Saturday, April 25, 2020

AI Meet & Greet

Hosted by
viraf and 2 others

Monday, April 27, 2020

Online: The COVID-19 Open Research Dataset

Hosted by
Seth Grimes

Quantum transfer learning

Quanvolutional Neural Networks

Monday, April 27, 2020

An introduction to Quantum Machine Learning

Hosted by
Debabrata Ghoshal

Machine Learning and AI: An Intuitive Introduction master class

Hosted by
Sri Krishnamurthy

Monday, April 27, 2020

AI-ML April 2020 Meeting : Overview of Microsoft AI Platform

Thursday, April 30, 2020

Remote Session: Neural Nets + Trees

Hosted by
Elsa Schaefer

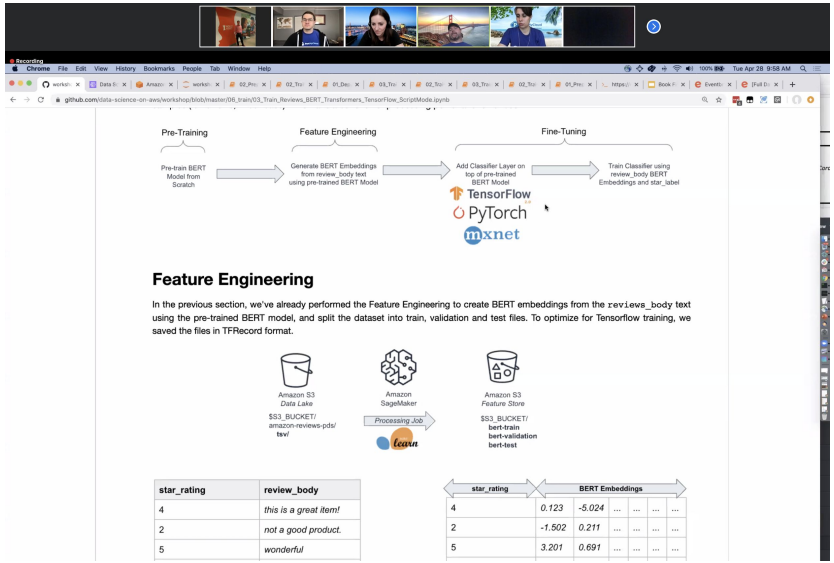
Source:
https://www.youtube.com/watch?v=zGFKSQlef_0
https://www.meetup.com/Bethesda-Artificial-Intelligence-Meetup/events/268561559/
https://www.meetup.com/DC-NLP/events/270026712/
https://www.meetup.com/Washington-Quantum-Computing-Meetup/events/269752061/
https://pennylane.ai/qml/demonstrations.html
https://www.meetup.com/QuantUniversity-Meetup-Washington-DC/events/270322193
https://www.dropbox.com/s/ncm4gfc0dkbl2bg/ML%20Master%20Class%20Final%20-%20Part%201-%20CFA%20Institute%202020.pdf?dl=0
https://www.meetup.com/ArtificialIntelligenceAndMachineLearning/events/268478285/
https://www.meetup.com/Machine-Learning-Paper-Club/events/270046053

Check out this new certification and new event

Tuesday, April 28, 2020

AWS DC-NOVA Virtual Meetup - Data Science and Machine Learning/AI on AWS

Hosted by
Chris Parlette and 2 others



TensorFlow Core

Introducing the TensorFlow Developer Certificate!

March 12, 2020

Posted by Alina Shinkarsky, on behalf of the TensorFlow Team

In the AI world today, more and more companies are looking to hire machine learning talent, and simultaneously, an increasing number of students and developers are looking for ways to gain and showcase their ML knowledge with formal recognition. In addition to the courses and learning resources available online, we want to help developers showcase their ML proficiency and help companies hire ML developers to solve challenging problems.

Data Science on Amazon Web Services
O'Reilly Book · Early 2021

Hands-on Learning with KubeFlow + GPU + Keras/TensorFlow 2.0 + TF Extended (TFX) + Kubernetes + SageMaker + PyTorch + XGBoost + Airflow + MLflow + Spark + Jupyter

Source:
https://blog.tensorflow.org/2020/03/introducing-tensorflow-developer-certificate.html?m=1
https://www.meetup.com/AWS-Washington-DC-Meet-Up/events/270065827/
https://www.eventbrite.com/e/full-day-workshop-kubeflow-gpu-kerastensorflow-20-tf-extended-tfx-kubernetes-pytorch-xgboost-tickets-63362929227
https://github.com/data-science-on-aws/workshop

Let’s keep it up and continue our TensorFlow journey 😀

FRI, MAY 1, 7:30 PM EDT

TensorFlow in Practice - Course 2 - Week 1,2 - Kaggle Challenge and...

Online event

Join us for our 4th adventure in Deep Learning! Just bring your curiosity and be ready to meet our growing community 😀 We are taking Course 2 of TensorFlow in Practice Specialization available at:...

39 attendees | Manage | Going

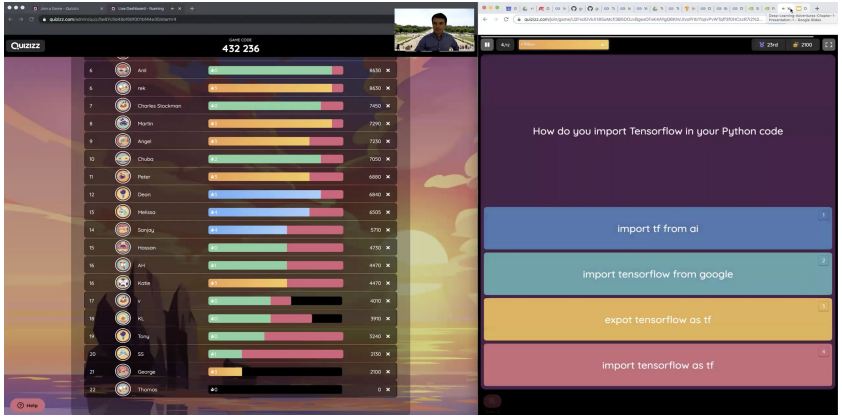
FRI, MAY 8, 7:30 PM EDT

TensorFlow in Practice-C2 Week 3,4- Transfer Learning & Multiclass...

Online event

Join us for our 5th adventure in Deep Learning! Just bring your curiosity and be ready to meet our growing community 😀 We are taking Course 2 of TensorFlow in Practice Specialization available at:...

42 attendees | 2 | Manage | Going



FRI, MAY 15, 7:30 PM EDT

Deep Learning Adventures - Happy Hour+Trivia Night 🍹🍸🥂

Online event

Join us for a fun conversation 🍹🍸🥂 No slides or recording this time, just getting to know each other better and forge a stronger community 😀 We are taking a break from our TensorFlow in Practice Specialization available at:...

18 attendees | Manage | Going

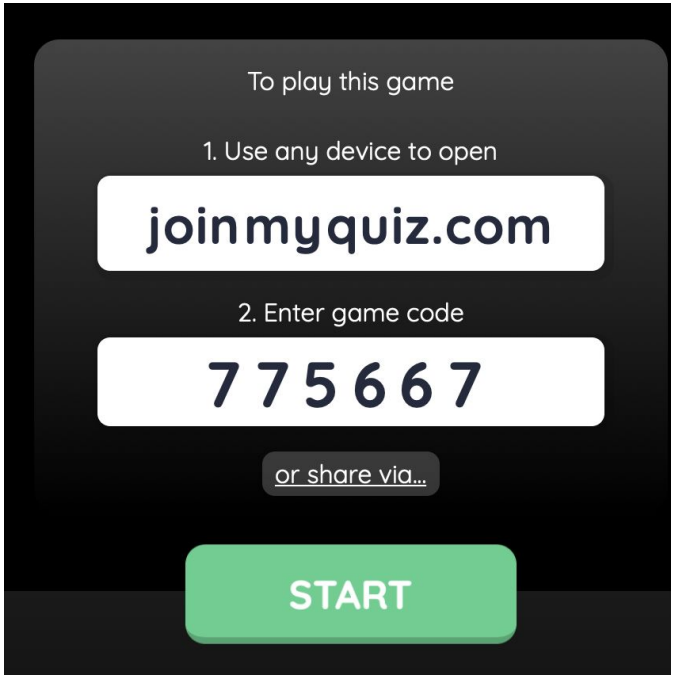
Source: https://www.meetup.com/Deep-Learning-Adventures

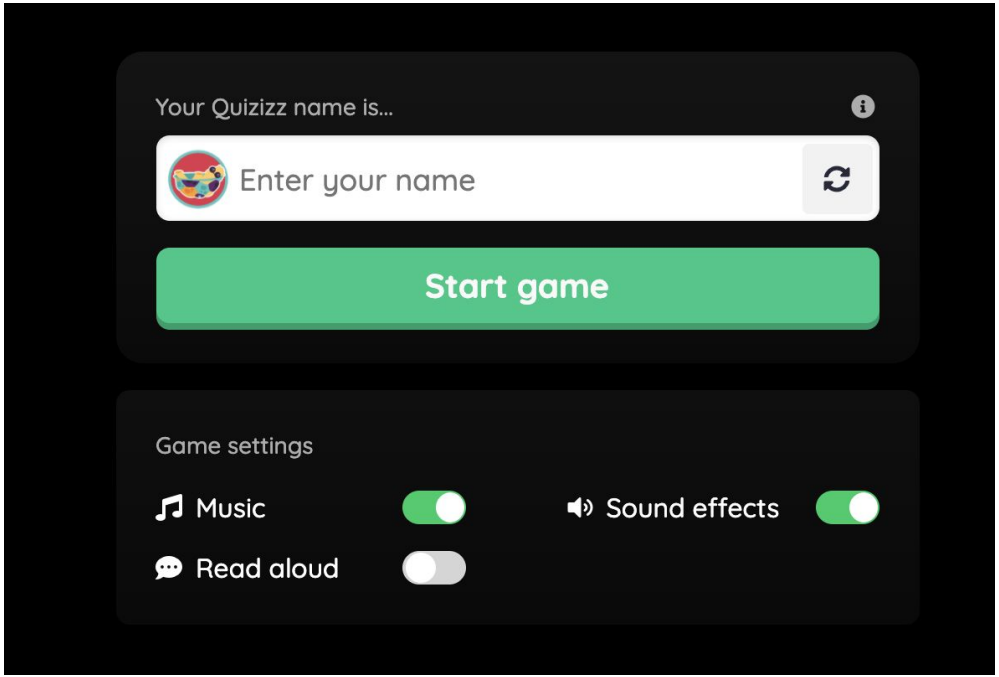
Time for a fun game 😀🎉

Deep Learning Adventures - Quiz 3

Professional Development · 1 times

Other

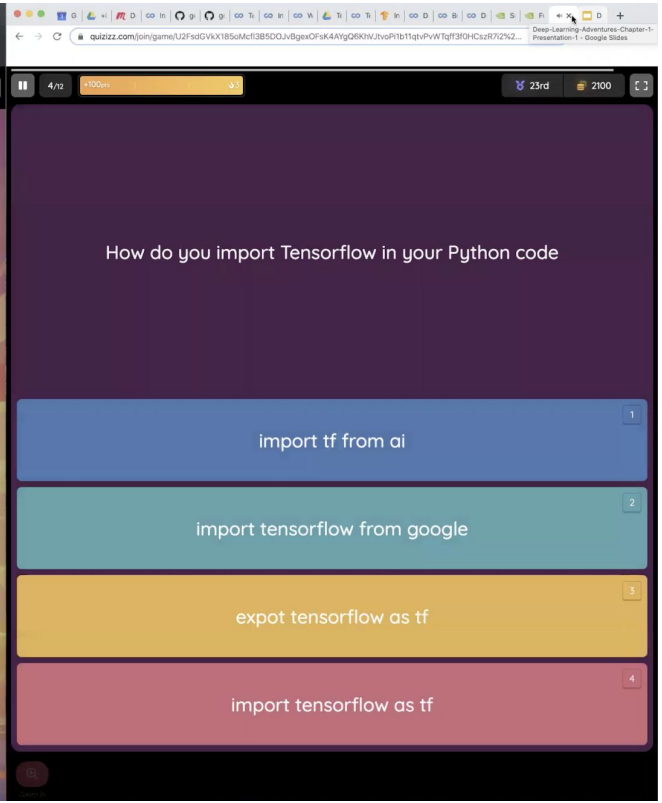




Practice here or use Flashcards:
https://quizizz.com/join/quiz/5eae10a807df73001b0c59a4/start?from=soloLinkShare&referrer=5d921444d0fa99001a135336

Time for a fun game 😀🎉

Practice here or use Flashcards:
https://quizizz.com/join/quiz/5eae10a807df73001b0c59a4/start?from=soloLinkShare&referrer=5d921444d0fa99001a135336



Questions

Discussion

Standard notations for Deep Learning

This document has the purpose of discussing a new standard for deep learning mathematical notations.

1 Neural Networks Notations.

General comments:

· superscript (i) will denote the i^{th} training example while superscript [l] will denote the l^{th} layer

Sizes:

·m : number of examples in the dataset

·n_x : input size

·n_y : output size (or number of classes)

·$n_h^{[l]}$: number of hidden units of the l^{th} layer

In a for loop, it is possible to denote $n_x = n_h^{[0]}$ and $n_y = n_h{}^{[\text{number of layers } +1]}$.

·L : number of layers in the network.

Objects:

·$X \in \mathbb{R}^{n_x \times m}$ is the input matrix

·$x^{(i)} \in \mathbb{R}^{n_x}$ is the i^{th}example represented as a column vector

·$Y \in \mathbb{R}^{n_y \times m}$ is the label matrix

·$y^{(i)} \in \mathbb{R}^{n_y}$ is the output label for the i^{th} example

·$W^{[l]} \in \mathbb{R}^{\text{number of units in next layer} \times \text{number of units in the previous layer}}$ is the weight matrix,superscript [l] indicates the layer

·$b^{[l]} \in \mathbb{R}^{\text{number of units in next layer}}$ is the bias vector in the l^{th} layer

·$\hat{y} \in \mathbb{R}^{n_y}$ is the predicted output vector. It can also be denoted $a^{[L]}$ where L is the number of layers in the network.

Common forward propagation equation examples:

$a = g^{[l]}(W_x x^{(i)} + b_1) = g^{[l]}(z_1)$ where $g^{[l]}$ denotes the l^{th} layer activation function

$\hat{y}^{(i)} = softmax(W_h h + b_2)$

· General Activation Formula: $a_j^{[l]} = g^{[l]}(\sum_k w_{jk}^{[l]} a_k^{[l-1]} + b_j^{[l]}) = g^{[l]}(z_j^{[l]})$

· $J(x, W, b, y)$ or $J(\hat{y}, y)$ denote the cost function.

Examples of cost function:

· $J_{CE}(\hat{y}, y) = -\sum_{i=0}^{m} y^{(i)} \log \hat{y}^{(i)}$

· $J_1(\hat{y}, y) = \sum_{i=0}^{m} \mid y^{(i)} - \hat{y}^{(i)} \mid$

2 Deep Learning representations

For representations:

· nodes represent inputs, activations or outputs

· edges represent weights or biases

Here are several examples of Standard deep learning representations

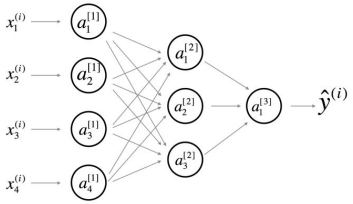


Figure 1: Comprehensive Network: representation commonly used for Neural Networks. For better aesthetic, we omitted the details on the parameters ($w_{ij}^{[l]}$ and $b_i^{[l]}$ etc...) that should appear on the edges

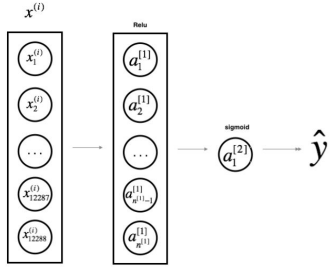


Figure 2: Simplified Network: a simpler representation of a two layer neural network, both are equivalent.