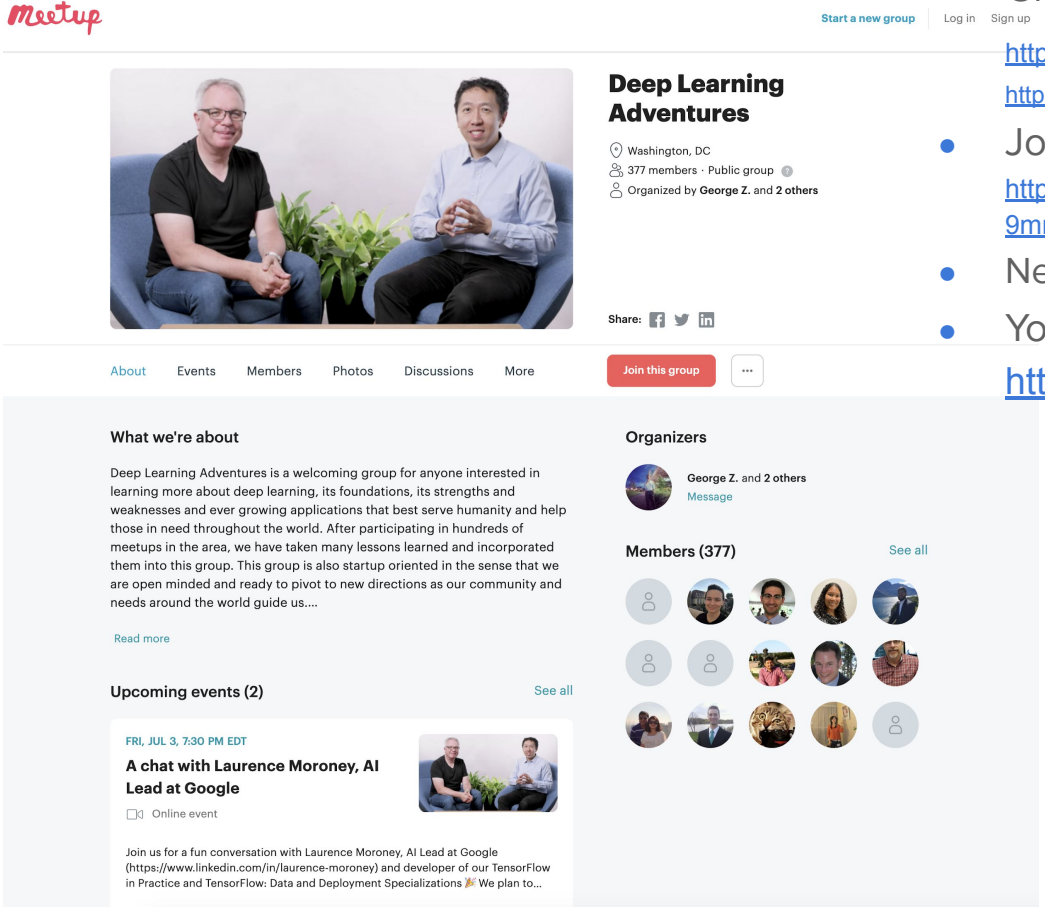
Deep Learning Adventures TensorFlow In Practice - Presentation 2

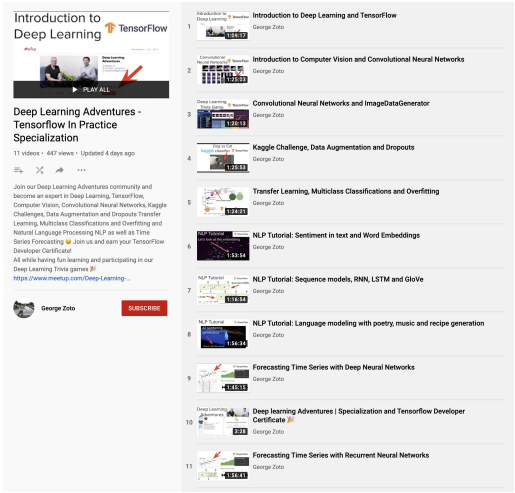
A quick overview of Coursera's Tensorflow in Practice specialization course

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

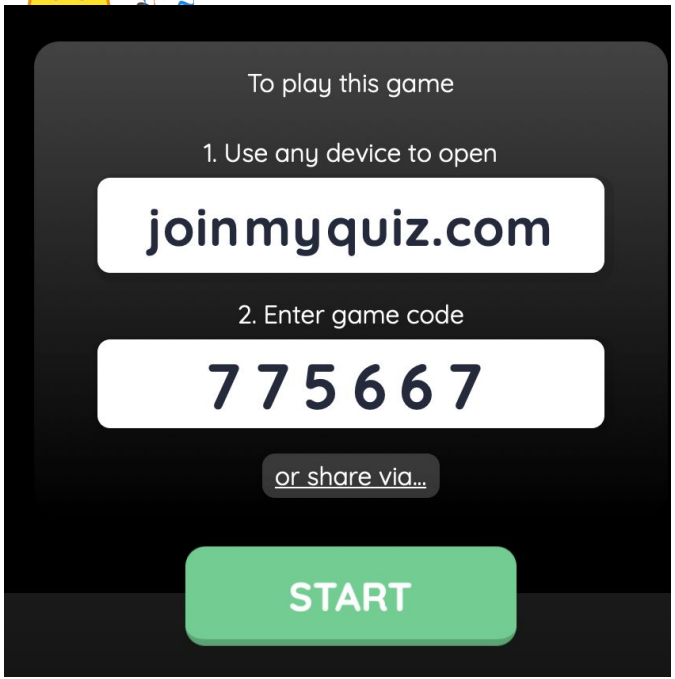
In the beginning...

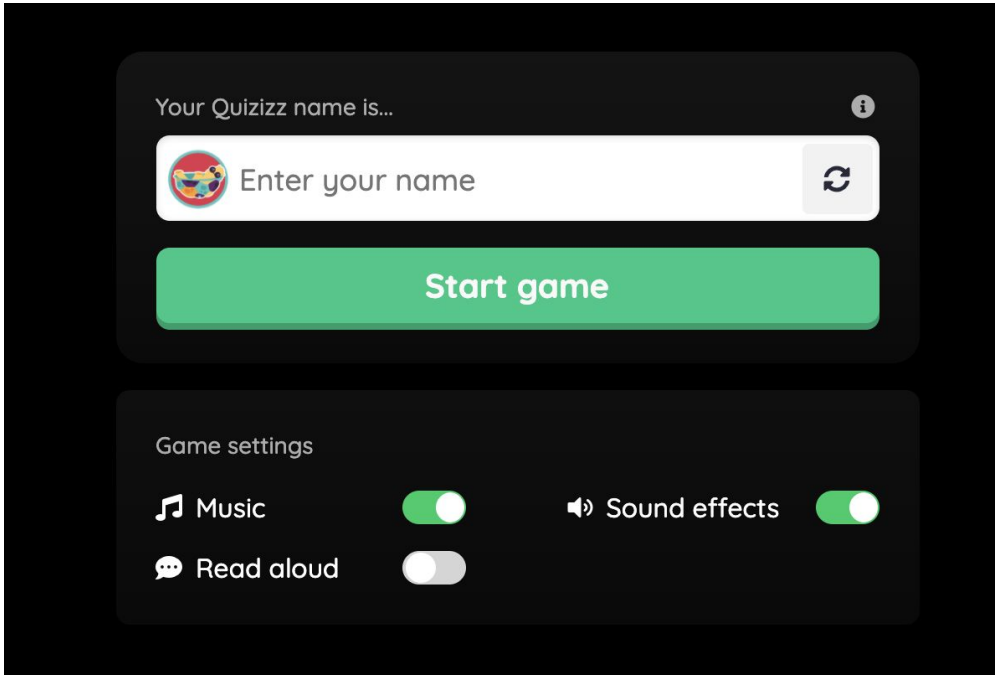
- Meetup Link
  https://www.meetup.com/Deep-Learning-Adventures
- GitHub repository
  https://github.com/georgezoto/Deep-Learning-Adventures
  https://github.com/georgezoto/TensorFlow-in-Practice
- Join us on Slack
  https://join.slack.com/t/deeplearninga-nmk8930/shared_invite/zt-d52h9mm9-h~Q0ZXw5PXsTDzPIINIvog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



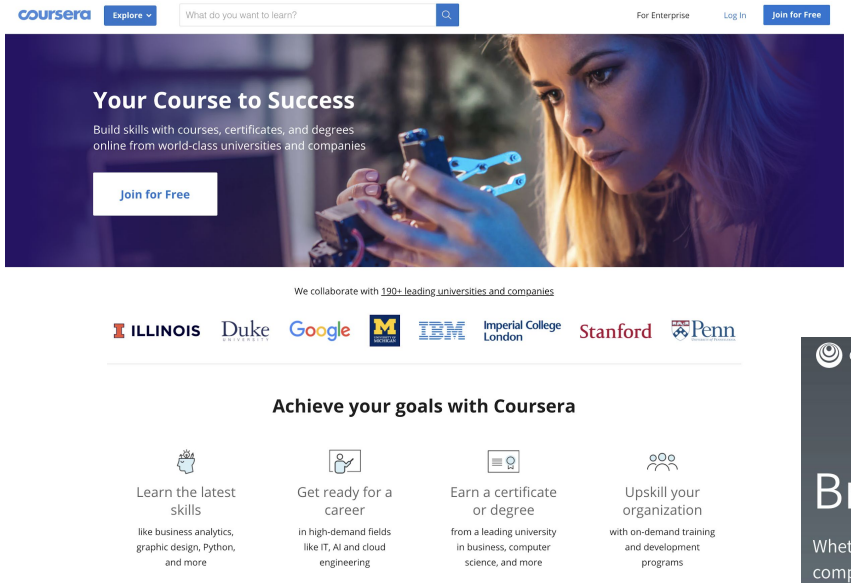


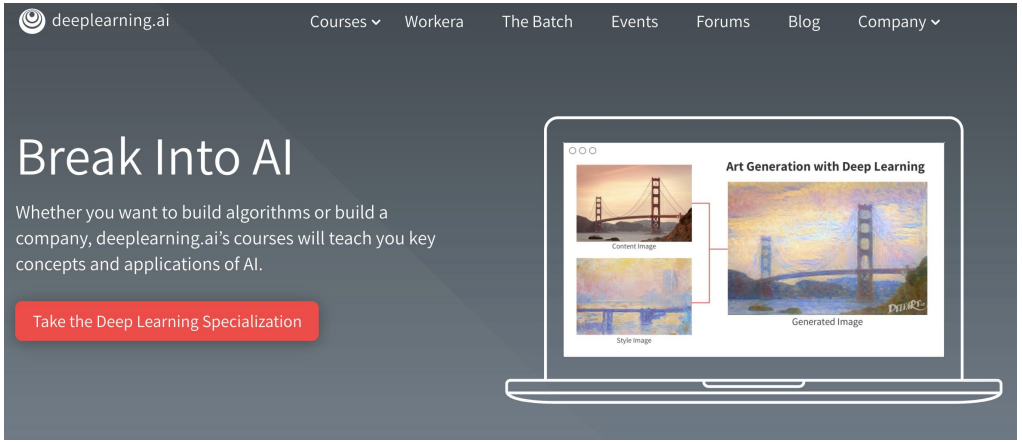
Not a typical Meetup... Get ready for a fun game 🥳





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

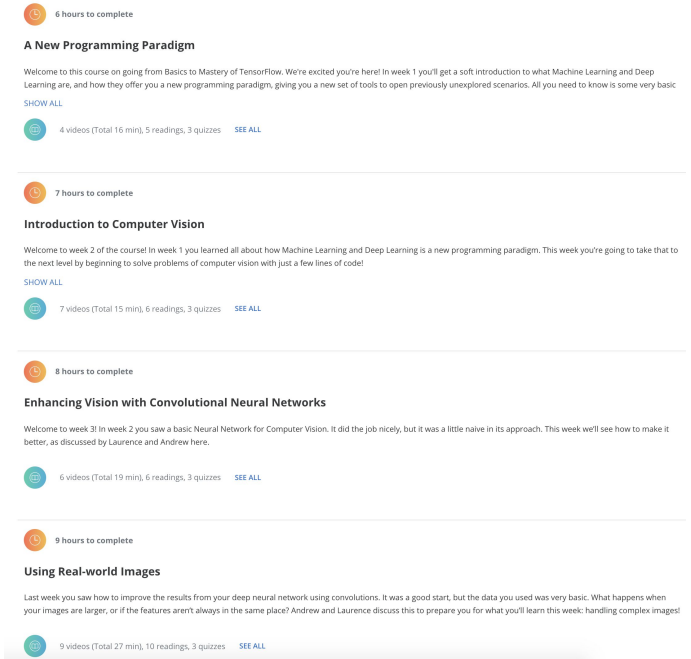
COURSE 4

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

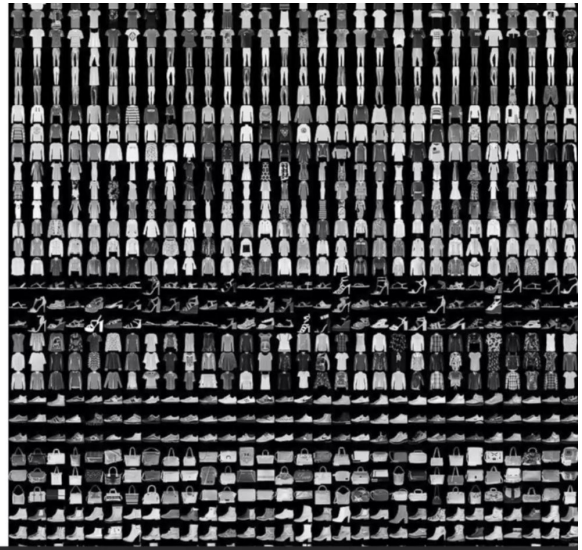
Course 1: Introduction to TensorFlow for Artificial Intelligence, Machine Learning, and Deep Learning

Week 1: A New Programming Paradigm (see Meetup #1)
Week 2: Introduction to Computer Vision
Week 3: Enhancing Vision with Convolutional Neural Networks
Week 4: Using Real-world Images

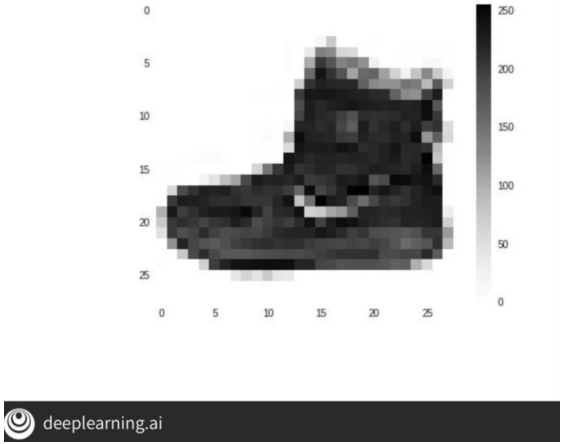
An Introduction to computer vision

Fashion MNIST

- 70k Images
- 10 Categories
- Images are 28x28
- Can train a neural net!

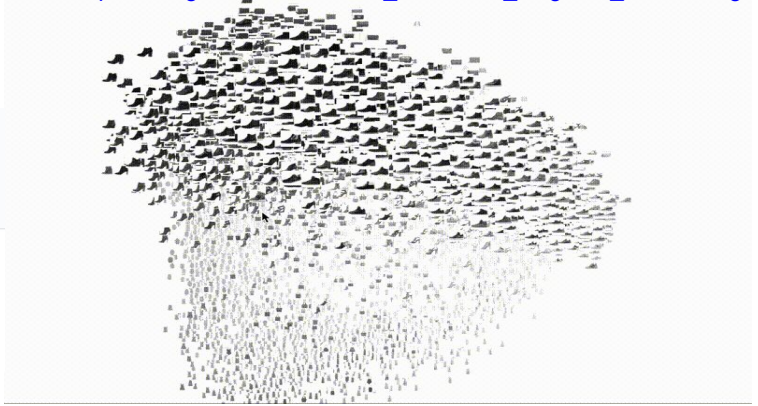


Label	Description
0	T-shirt/top
1	Trouser
2	Pullover
3	Dress
4	Coat
5	Sandal
6	Shirt
7	Sneaker
8	Bag
9	Ankle boot

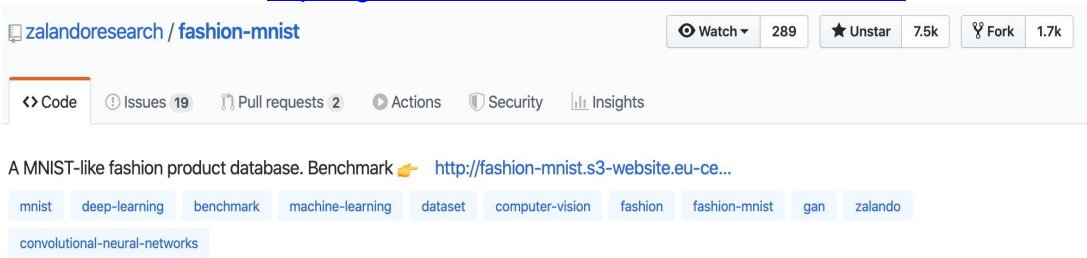


Visualization: t-SNE

https://en.wikipedia.org/wiki/T-distributed_stochastic_neighbor_embedding



Fashion MNIST: https://github.com/zalandoresearch/fashion-mnist



An Introduction to computer vision

Loss functions:
https://keras.io/losses/
https://www.tensorflow.org/api_docs/python/tf/keras/losses/Loss
http://wiki.fast.ai/index.php/Log_Loss
https://gombru.github.io/2018/05/23/cross_entropy_loss/

```
import tensorflow as tf
print(tf.__version__)
from tensorflow import keras
```

```
mnist = tf.keras.datasets.fashion_mnist
```

When using the `sparse_categorical_crossentropy` loss, your targets should be *integer targets*. If you have categorical targets, you should use `categorical_crossentropy`.

`categorical_crossentropy` is another term for multi-class log loss.

```
mnist
```

```
<module 'tensorflow.keras.datasets.fashion_mnist' from '/usr/local/lib/python3.6/dist-packages/tensorflow/keras/datasets/fashion_mnist/__init__.py'>
```

```
(training_images, training_labels), (test_images, test_labels) = mnist.load_data()
```

```
training_images  = training_images / 255.0
test_images = test_images / 255.0
```

https://www.tensorflow.org/api_docs/python/tf/keras/Model#compile

```
model = tf.keras.models.Sequential([tf.keras.layers.Flatten(input_shape=(28,28)),
                                    tf.keras.layers.Dense(128, activation=tf.nn.relu),
                                    tf.keras.layers.Dense(10, activation=tf.nn.softmax)])
```

```
model.compile(optimizer = tf.optimizers.Adam(),
              loss = 'sparse_categorical_crossentropy',
              metrics=['accuracy'])
```

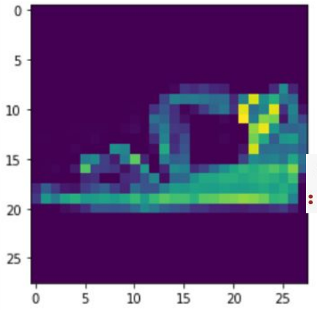
```
model.fit(training_images, training_labels, epochs=5)
```

```
Epoch 1/5
1875/1875 [==============================] - 4s 2ms/step - loss: 0.4913 - accuracy: 0.8274
Epoch 2/5
1875/1875 [==============================] - 4s 2ms/step - loss: 0.3750 - accuracy: 0.8650
Epoch 3/5
1875/1875 [==============================] - 4s 2ms/step - loss: 0.3350 - accuracy: 0.8772
Epoch 4/5
1875/1875 [==============================] - 4s 2ms/step - loss: 0.3139 - accuracy: 0.8855
Epoch 5/5
1875/1875 [==============================] - 4s 2ms/step - loss: 0.2933 - accuracy: 0.8916
```

An Introduction to computer vision

```
np.random.seed(104990)
dex = np.random.randint(0,60000)
print(dex)
print(training_labels[dex])
print(training_images[dex])
plt.imshow(training_images[dex])
```

```
2042
5
[[  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   5  39  31  42  29  24   0   0   0   0  31 124  27   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0  53 111 102 106  98  89  91  31 111 240 126 160  98  87   0]
 [  0   0   0   0   0   0   0   0   0   0   1   0  39  68  35   3  14  46  81 100  39 245  91 115 245  37 115  13]
 [  0   0   0   0   0   0   0   0   0   0   3   0  94  81  37  11   0   0   0   5   0 255 104 210 202  14 102  53]
 [  0   0   0   0   1   1   1   1   0   3   7   0  67 104  42  39   0   0   0   0   0   0 214 249 134  85  81  87]
 [  1   1   0   0   0   1   3   0   0   0   0   0  33 113  31  61   0   0   0   3   0   0 202 156  74 132 139  87]
 [  0   0   0   0   3   0   0   0 128 117   0   0   7 156  37  67   0   0   0   0   0  13 238  91  94  89  93  76]
 [  0   0   0   0   0 107 122  48  20 106 193  27   0 109  76  80   0   0   0   0   0  48 122 109 115  98 152  93]
 [  0   1   3   0  18 187  52  46  39  11  39 137   0  29 120 111  33  52  89 124 150 167 158 171 186 169 208  80]
 [  0   0   0   0   0  29  35   0   0  31  35  65 117   9  85  87 130 152 152 161 160 154 160 167 160 147 176  13]
 [ 29  76  42  18  14  46  63  80  46  27  91  67  67 113 141 139 148 145 143 148 145 143 148 156 145 158 143   0]
 [ 42 134 104 147 161 137 141 148 141 152 171 141 171 189 178 178 163 189 191 202 204 212 208 197 167 171 113   0]
 [  0   0   0   9  16  33  67  67  65  67  89 119 102  78  67  59  39  39  29  27  22  22  29  27  37  44  22   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0]]
```




```
training_history = model.fit
```

```
np.average(training_history.history['loss']))
: ", np.average(training_history.history['accuracy']))
```

```
Average training loss:  0.361683636903762 8
Average training accuracy:  0.8693400025367737
```

```
model.evaluate(test_images, test_labels)
```

```
313/313 [==============================] - 1s 3ms/step - loss: 0.0748
```

```
print(classifications[0])
```

```
[8.2633102e-11 1.0370549e-09 1.1534371e-09 4.8558093e-05 2.9175582e-14
 9.6861207e-11 3.0065261e-15 9.9995136e-01 5.7236882e-10 1.3707833e-07]
```

```
print(test_labels[0])
```

7

```
model.summary()
```

```
Model: "sequential_4"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
flatten_4 (Flatten)          (None, 784)               0
_________________________________________________________________
dense_8 (Dense)              (None, 1024)              803840
_________________________________________________________________
dense_9 (Dense)              (None, 10)                10250
=================================================================
Total params: 814,090
Trainable params: 814,090
Non-trainable params: 0
_________________________________________________________________
```

An Introduction to computer vision

```
import tensorflow as tf
print(tf.__version__)

mnist = tf.keras.datasets.mnist

(training_images, training_labels) ,  (test_images, test_labels) = mnist.load_data()

training_images = training_images/255.0
test_images = test_images/255.0

model = tf.keras.models.Sequential([tf.keras.layers.Flatten(input_shape=(28,28)),
                                    tf.keras.layers.Dense(128, activation=tf.nn.relu),
                                    tf.keras.layers.Dense(10, activation=tf.nn.softmax)])

model.compile(optimizer = 'adam',
              loss = 'sparse_categorical_crossentropy')

model.fit(training_images, training_labels, epochs=5)

model.evaluate(test_images, test_labels)

classifications = model.predict(test_images)

print(classifications[0])

print(test_labels[0])
```

- Experiment with different values for the dense layer size:

Test Data:

Size	Loss	Acc	Time (s)
128	0.343	0.875	27
512	0.078	0.977	36
1024	0.075	0.978	50
4096	0.084	0.978	91

What would happen if:

- Flatten() layer removed
- Changed number of outputs (i.e. fewer than 10)
- More layers (deeper network)
- More epochs (train longer)
- Remove normalization (/255)

An Introduction to computer vision

```
class myCallback(tf.keras.callbacks.Callback):
  def on_epoch_end(self, epoch, logs={}):
    if(logs.get('loss')<0.4):
      print("\nReached 60% accuracy so cancelling training!")
      self.model.stop_training = True

callbacks = myCallback()
```

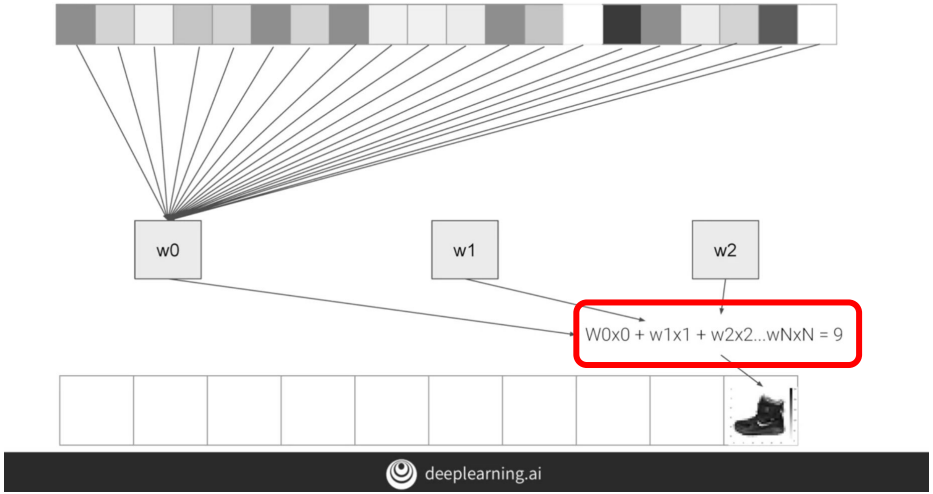
- How to stop training at a specific loss or accuracy threshold?

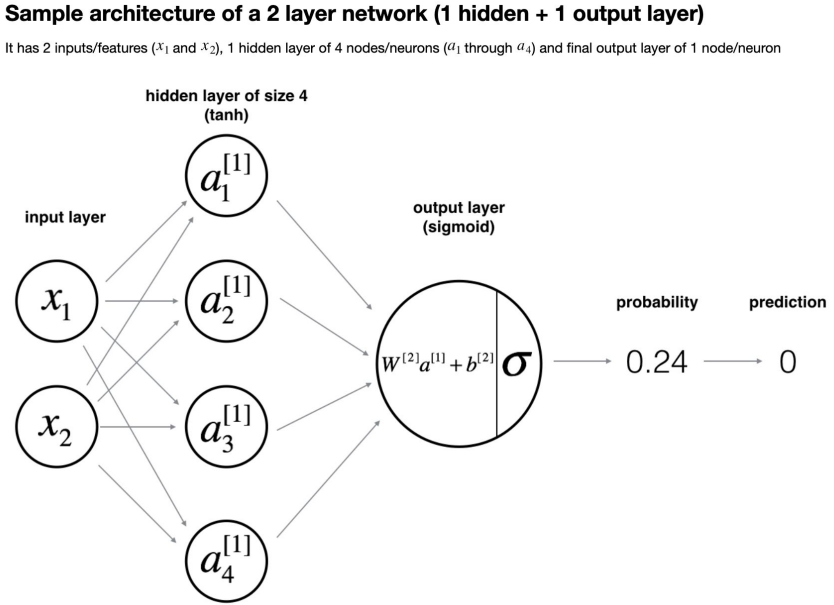
```
model.fit(training_images, training_labels, epochs=5, callbacks=[callbacks])
```

```
2.2.0-rc2
Epoch 1/5
1875/1875 [==============================] - 8s 4ms/step - loss: 0.4755
Epoch 2/5
1871/1875 [============================>.] - ETA: 0s - loss: 0.3617
Reached 60% accuracy so cancelling training!
1875/1875 [==============================] - 8s 4ms/step - loss: 0.3616
<tensorflow.python.keras.callbacks.History at 0x7fb03aab6e48>
```

An Introduction to computer vision

```
import tensorflow as tf
print(tf.__version__)

mnist = tf.keras.datasets.mnist

(training_images, training_labels) ,  (test_images, test_labels) = mnist.load_data()

training_images = training_images/255.0
test_images = test_images/255.0

model = tf.keras.models.Sequential([tf.keras.layers.Flatten(input_shape=(28,28)),
                                    tf.keras.layers.Dense(128, activation=tf.nn.relu),
                                    tf.keras.layers.Dense(10, activation=tf.nn.softmax)])
```





Andrew Ng: https://youtu.be/fXOsFF95ifk

An Introduction to computer vision

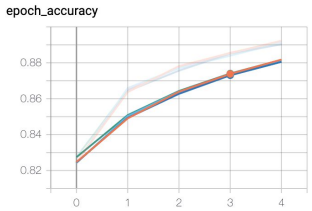
TensorBoard

https://colab.research.google.com/github/tensorflow/tensorboard/blob/master/docs/scalars_and_keras.ipynb

```
# Define the Keras TensorBoard callback.
logdir = "logs/scalars/" + datetime.now().strftime("%Y%m%d-%H%M%S")
tensorboard_callback = keras.callbacks.TensorBoard(log_dir=logdir)
```

```
training_history = model.fit(training_images,
          training_labels,
          #batch_size = 64,
          #verbose = 0,
          epochs = 5,
          #validation_data = (test_images, test_labels),
          callbacks=[tensorboard_callback])
```

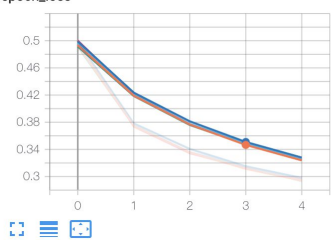
epoch_accuracy



Name	Smoothed	Value	Step	Time	Relative
20200410-165543/train	0.8738	0.8854	3	Fri Apr 10, 12:56:03	13s
20200410-171047/train	0.873	0.8852	3	Fri Apr 10, 13:11:06	12s
20200410-171729/train	0.8731	0.884	3	Fri Apr 10, 13:17:47	12s
20200410-175850/train	0.8738	0.8855	3	Fri Apr 10, 13:59:09	12s
20200410-180421/train	0.8741	0.8855	3	Fri Apr 10, 14:04:39	12s

Name	Smoothed	Value	Step	Time	Relative
20200410-165543/train	0.3466	0.3113	3	Fri Apr 10, 12:56:03	13s
20200410-171047/train	0.3509	0.3152	3	Fri Apr 10, 13:11:06	12s
20200410-171729/train	0.3489	0.3152	3	Fri Apr 10, 13:17:47	12s
20200410-175850/train	0.3475	0.3128	3	Fri Apr 10, 13:59:09	12s
20200410-180421/train	0.3474	0.3139	3	Fri Apr 10, 14:04:39	12s

epoch_loss

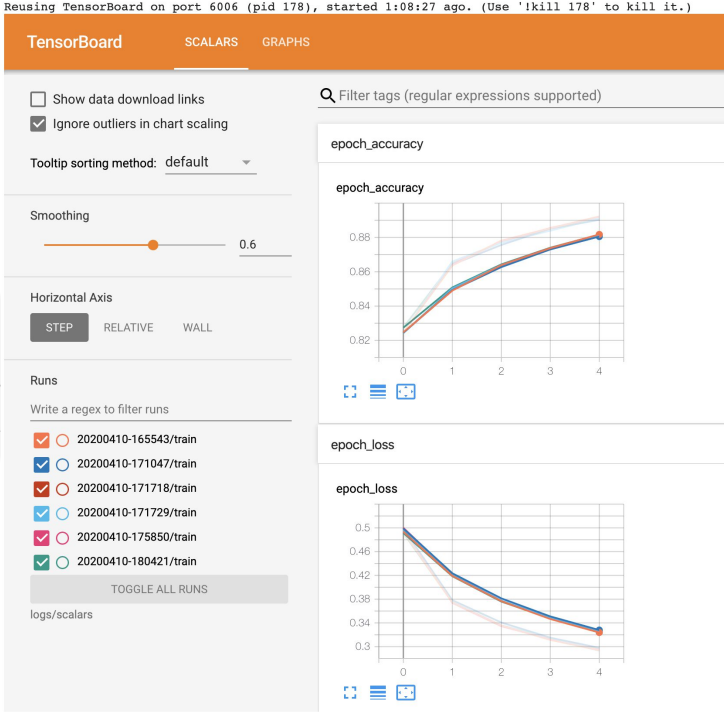


Op-level graph

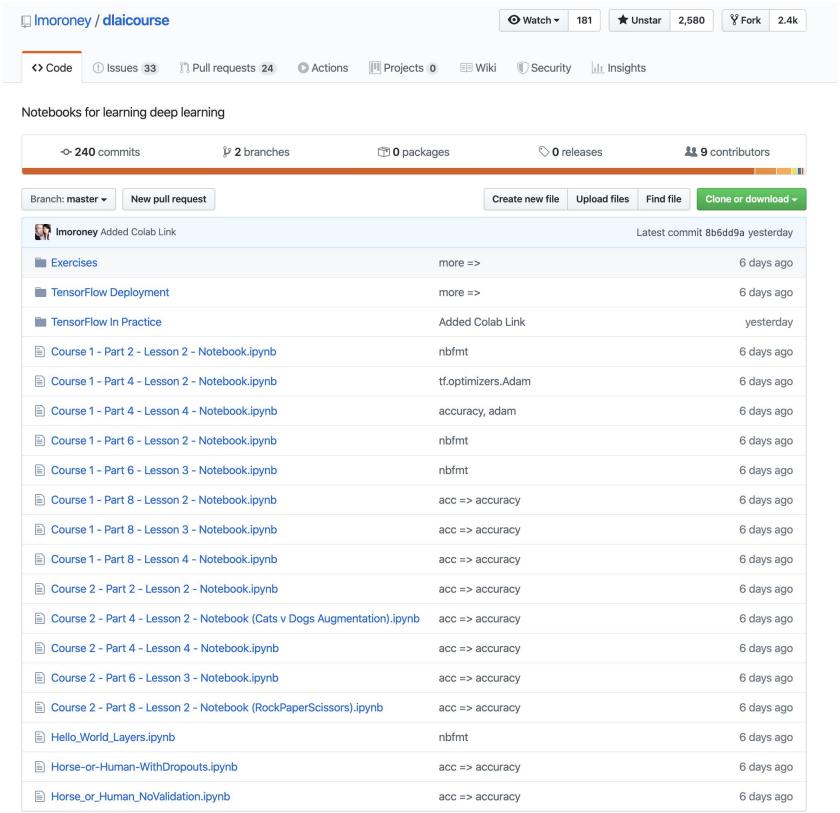
Start TensorBoard and wait a few seconds for the UI to load. Select the Graphs dashboard by tapping "Graphs" at the top.

```
[15] %tensorboard --logdir logs/scalars
```

```
Reusing TensorBoard on port 6006 (pid 178), started 1:08:27 ago. (Use '!kill 178' to kill it.)
```



Laurence's GitHub repository



lmoroney / dlaicourse

Watch 181 | Unstar 2,580 | Fork 2.4k

Code | Issues 33 | Pull requests 24 | Actions | Projects 0 | Wiki | Security | Insights

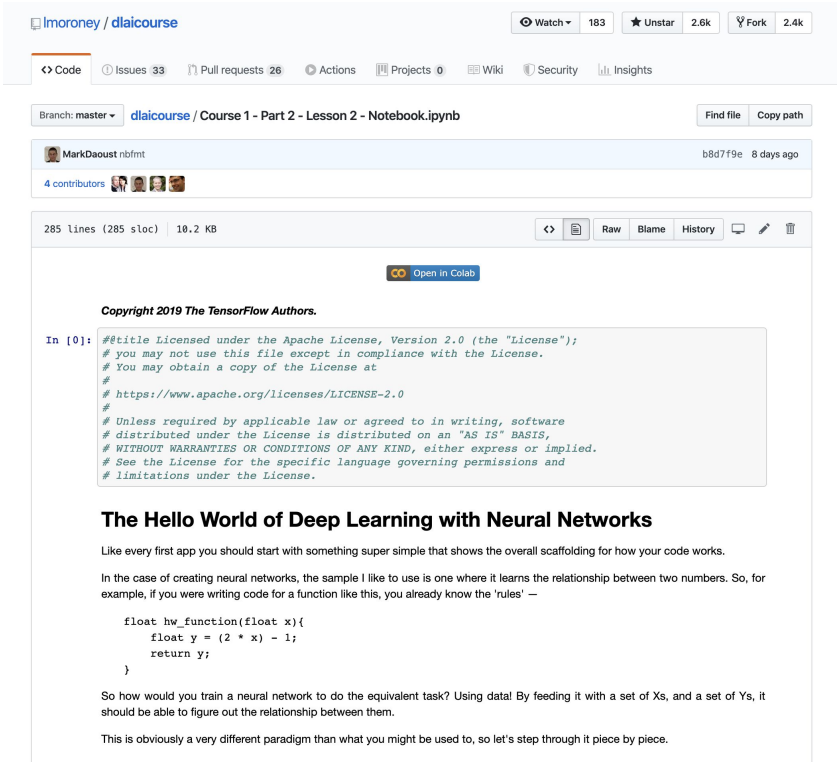
Notebooks for learning deep learning

240 commits | 2 branches | 0 packages | 0 releases | 9 contributors

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

lmoroney Added Colab Link — Latest commit 8b6dd9a yesterday

Exercises	more =>	6 days ago
TensorFlow Deployment	more =>	6 days ago
TensorFlow In Practice	Added Colab Link	yesterday
Course 1 - Part 2 - Lesson 2 - Notebook.ipynb	nbfmt	6 days ago
Course 1 - Part 4 - Lesson 2 - Notebook.ipynb	tf.optimizers.Adam	6 days ago
Course 1 - Part 4 - Lesson 4 - Notebook.ipynb	accuracy, adam	6 days ago
Course 1 - Part 6 - Lesson 2 - Notebook.ipynb	nbfmt	6 days ago
Course 1 - Part 6 - Lesson 3 - Notebook.ipynb	nbfmt	6 days ago
Course 1 - Part 8 - Lesson 2 - Notebook.ipynb	acc => accuracy	6 days ago
Course 1 - Part 8 - Lesson 3 - Notebook.ipynb	acc => accuracy	6 days ago
Course 1 - Part 8 - Lesson 4 - Notebook.ipynb	acc => accuracy	6 days ago
Course 2 - Part 2 - Lesson 2 - Notebook.ipynb	acc => accuracy	6 days ago
Course 2 - Part 4 - Lesson 2 - Notebook (Cats v Dogs Augmentation).ipynb	acc => accuracy	6 days ago
Course 2 - Part 4 - Lesson 4 - Notebook.ipynb	acc => accuracy	6 days ago
Course 2 - Part 6 - Lesson 3 - Notebook.ipynb	acc => accuracy	6 days ago
Course 2 - Part 8 - Lesson 2 - Notebook (RockPaperScissors).ipynb	acc => accuracy	6 days ago
Hello_World_Layers.ipynb	nbfmt	6 days ago
Horse-or-Human-WithDropouts.ipynb	acc => accuracy	6 days ago
Horse_or_Human_NoValidation.ipynb	acc => accuracy	6 days ago



lmoroney / dlaicourse

Watch 183 | Unstar 2.6k | Fork 2.4k

Code | Issues 33 | Pull requests 26 | Actions | Projects 0 | Wiki | Security | Insights

Branch: master | dlaicourse / Course 1 - Part 2 - Lesson 2 - Notebook.ipynb | Find file | Copy path

MarkDaoust nbfmt — b8d7f9e 8 days ago

4 contributors

285 lines (285 sloc) | 10.2 KB | Raw | Blame | History

Open in Colab

Copyright 2019 The TensorFlow Authors.

In [0]:

```
#@title Licensed under the Apache License, Version 2.0 (the "License");
# you may not use this file except in compliance with the License.
# You may obtain a copy of the License at
#
# https://www.apache.org/licenses/LICENSE-2.0
#
# Unless required by applicable law or agreed to in writing, software
# distributed under the License is distributed on an "AS IS" BASIS,
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
# See the License for the specific language governing permissions and
# limitations under the License.
```

The Hello World of Deep Learning with Neural Networks

Like every first app you should start with something super simple that shows the overall scaffolding for how your code works.

In the case of creating neural networks, the sample I like to use is one where it learns the relationship between two numbers. So, for example, if you were writing code for a function like this, you already know the 'rules' —

```
float hw_function(float x){
    float y = (2 * x) - 1;
    return y;
}
```

So how would you train a neural network to do the equivalent task? Using data! By feeding it with a set of Xs, and a set of Ys, it should be able to figure out the relationship between them.

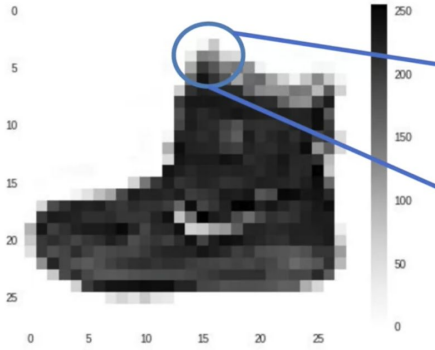
This is obviously a very different paradigm than what you might be used to, so let's step through it piece by piece.

Source: https://github.com/lmoroney/dlaicourse

Course 1: Introduction to TensorFlow for Artificial Intelligence, Machine Learning, and Deep Learning

Week 1: A New Programming Paradigm (see Meetup #1)
Week 2: Introduction to Computer Vision
Week 3: Enhancing Vision with Convolutional Neural Networks
Week 4: Using Real-world Images

Enhancing Vision with Convolutional Neural Networks



0	64	128
48	192	144
142	226	168

Current Pixel Value is 192

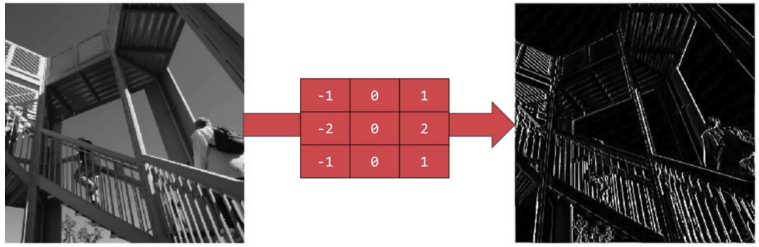
Consider neighbor Values

-1	0	-2
.5	4.5	-1.5
1.5	2	-3

Filter Definition

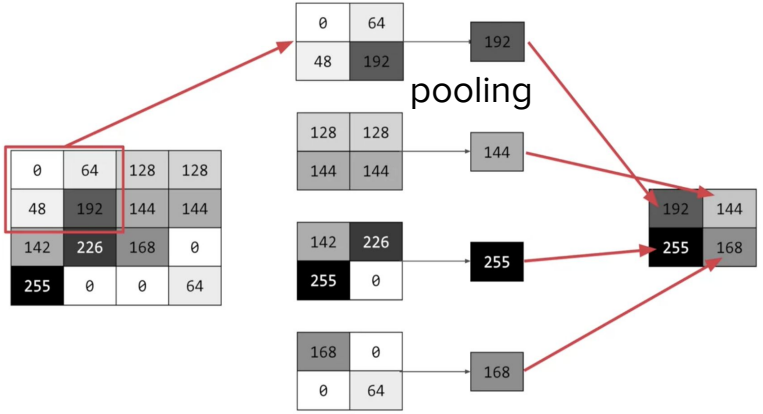
cross-correlation

```
CURRENT_PIXEL_VALUE = 192
NEW_PIXEL_VALUE = (-1 * 0) + (0 * 64) + (-2 * 128) +
                  (.5 * 48) + (4.5 * 192) + (-1.5 * 144) +
                  (1.5 * 42) + (2 * 226) + (-3 * 168)
```



deeplearning.ai

deeplearning.ai



deeplearning.ai

Enhancing Vision with Convolutional Neural Networks

```
model = tf.keras.models.Sequential([
  tf.keras.layers.Conv2D(64, (3,3), activation='relu',
                         input_shape=(28, 28, 1)),
  tf.keras.layers.MaxPooling2D(2, 2),
  tf.keras.layers.Conv2D(64, (3,3), activation='relu'),
  tf.keras.layers.MaxPooling2D(2,2),
  tf.keras.layers.Flatten(),
  tf.keras.layers.Dense(128, activation='relu'),
  tf.keras.layers.Dense(10, activation='softmax')
])
```

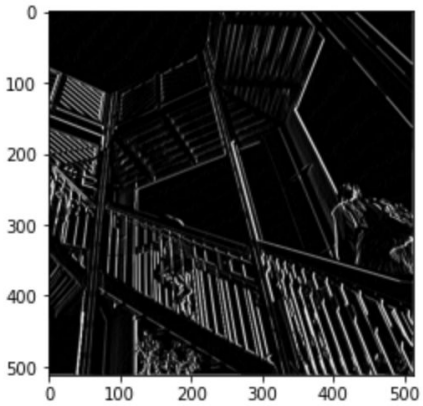
```
model = tf.keras.models.Sequential([
  tf.keras.layers.Conv2D(64, (3,3), activation='relu',
                         input_shape=(28, 28, 1)),
  tf.keras.layers.MaxPooling2D(2, 2),
  tf.keras.layers.Conv2D(64, (3,3), activation='relu'),
  tf.keras.layers.MaxPooling2D(2,2),
  tf.keras.layers.Flatten(),
  tf.keras.layers.Dense(128, activation='relu'),
  tf.keras.layers.Dense(10, activation='softmax')
])
```

```
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_12 (Conv2D)           (None, 26, 26, 64)        640
_________________________________________________________________
max_pooling2d_12 (MaxPooling (None, 13, 13, 64)        0
_________________________________________________________________
conv2d_13 (Conv2D)           (None, 11, 11, 64)        36928
_________________________________________________________________
max_pooling2d_13 (MaxPooling (None, 5, 5, 64)          0
_________________________________________________________________
flatten_5 (Flatten)          (None, 1600)              0
_________________________________________________________________
dense_10 (Dense)             (None, 128)               204928
_________________________________________________________________
dense_11 (Dense)             (None, 10)                1290
=================================================================
```

Enhancing Vision with Convolutional Neural Networks

```
# This filter detects edges nicely
# It creates a convolution that only passes through sharp edges and straight
# lines.

#Experiment with different values for fun effects.
#filter = [ [0, 1, 0], [1, -4, 1], [0, 1, 0]]

# A couple more filters to try for fun!
filter = [ [-1, -2, -1], [0, 0, 0], [1, 2, 1]]
#filter = [ [-1, 0, 1], [-2, 0, 2], [-1, 0, 1]]

# If all the digits in the filter don't add up to 0 or 1, you
# should probably do a weight to get it to do so
# so, for example, if your weights are 1,1,1 1,2,1 1,1,1
# They add up to 10, so you would set a weight of .1 if you want to normalize them
weight  = 1
```

```
for x in range(1,size_x-1):
  for y in range(1,size_y-1):
      convolution = 0.0
      convolution = convolution + (i[x - 1, y-1] * filter[0][0])
      convolution = convolution + (i[x, y-1] * filter[0][1])
      convolution = convolution + (i[x + 1, y-1] * filter[0][2])
      convolution = convolution + (i[x-1, y] * filter[1][0])
      convolution = convolution + (i[x, y] * filter[1][1])
      convolution = convolution + (i[x+1, y] * filter[1][2])
      convolution = convolution + (i[x-1, y+1] * filter[2][0])
      convolution = convolution + (i[x, y+1] * filter[2][1])
      convolution = convolution + (i[x+1, y+1] * filter[2][2])
      convolution = convolution * weight
      if(convolution<0):
        convolution=0
      if(convolution>255):
        convolution=255
      i_transformed[x, y] = convolution
```

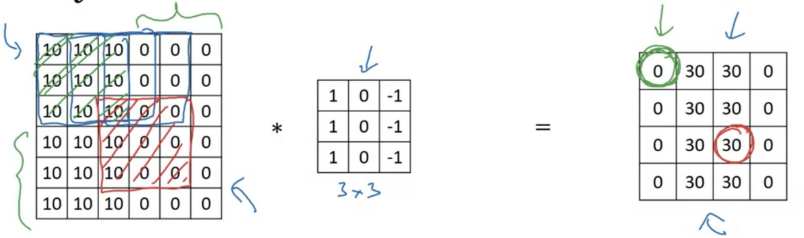


https://lodev.org/cgtutor/filtering.html

Enhancing Vision with Convolutional Neural Networks

Why convolutions

translation invariance



Parameter sharing: A feature detector (such as a vertical edge detector) that's useful in one part of the image is probably useful in another part of the image.

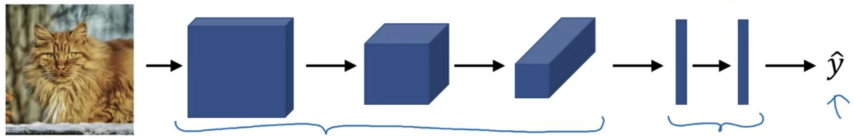
→ **Sparsity of connections:** In each layer, each output value depends only on a small number of inputs.

Andrew Ng

Putting it together

Training set $(x^{(1)}, y^{(1)}) \ldots (x^{(m)}, y^{(m)})$.

w, b



Cost $J = \frac{1}{m} \sum_{i=1}^{m} \mathcal{L}(\hat{y}^{(i)}, y^{(i)})$

Use gradient descent to optimize parameters to reduce J

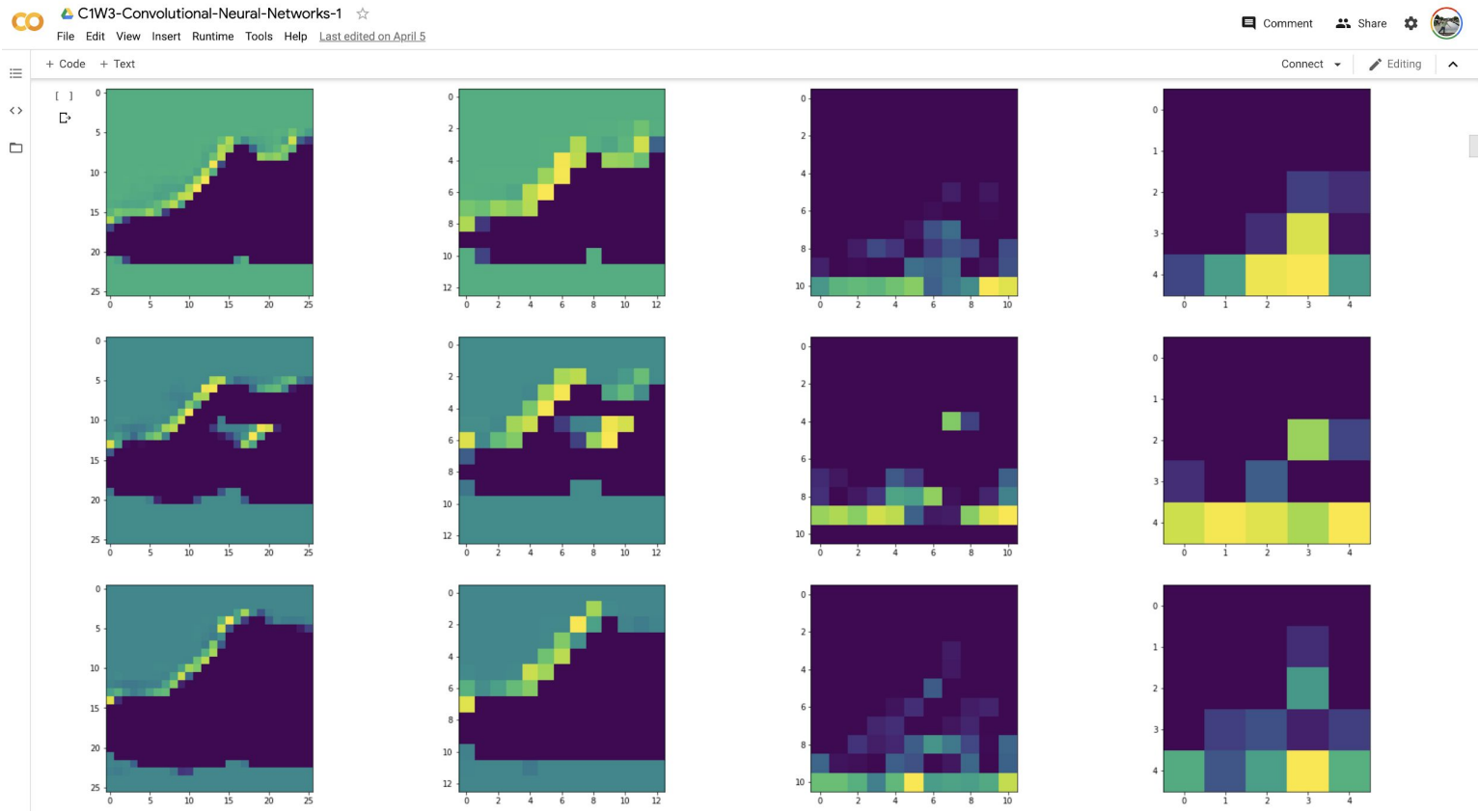
Andrew Ng

Source: https://www.coursera.org/lecture/convolutional-neural-networks/why-convolutions-Xv7B5

As you can see we go from the raw pixels of the images to increasingly abstract and compact representations. The representations downstream start highlighting what the network pays attention to, and they show fewer and fewer features being "activated"; most are set to zero. This is called "sparsity." Representation sparsity is a key feature of deep learning.

These representations carry increasingly less information about the original pixels of the image, but increasingly refined information about the class of the image. You can think of a convnet (or a deep network in general) as an information distillation pipeline.

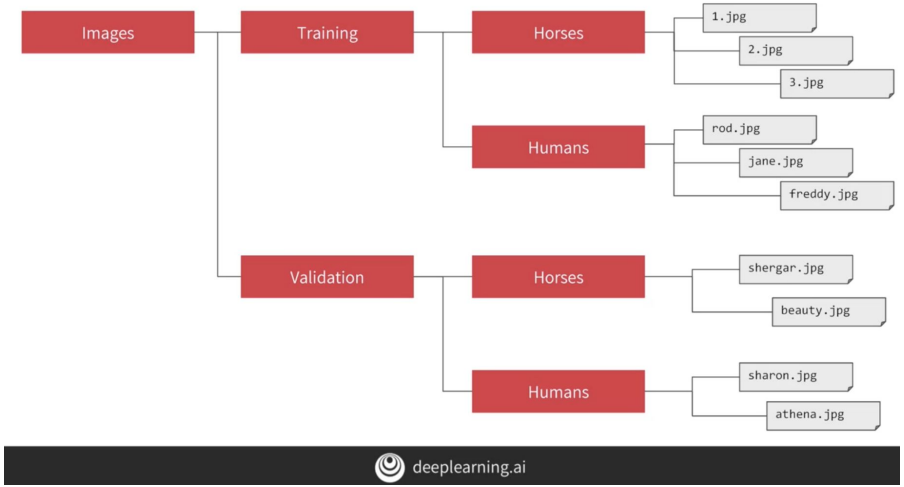
Enhancing Vision with Convolutional Neural Networks - Colab



Course 1: Introduction to TensorFlow for Artificial Intelligence, Machine Learning, and Deep Learning

Week 1: A New Programming Paradigm (see Meetup #1)
Week 2: Introduction to Computer Vision
Week 3: Enhancing Vision with Convolutional Neural Networks
Week 4: Using Real-world Images

Using Real-world Images




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

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

train_generator = train_datagen.flow_from_directory(
        train_dir,
        target_size=(300, 300),
        batch_size=128,
        class_mode='binary')
```

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

validation_generator = test_datagen.flow_from_directory(
        validation_dir,
        target_size=(300, 300),
        batch_size=32,
        class_mode='binary')
```

Using Real-world Images

1

```
model = tf.keras.models.Sequential([
    # Note the input shape is the desired size of the image 300x300 with 3 bytes color
    # This is the first convolution
    tf.keras.layers.Conv2D(16, (3,3), activation='relu', input_shape=(300, 300, 3)),
    tf.keras.layers.MaxPooling2D(2, 2),
    # The second convolution
    tf.keras.layers.Conv2D(32, (3,3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2,2),
    # The third convolution
    tf.keras.layers.Conv2D(64, (3,3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2,2),
    # The fourth convolution
    tf.keras.layers.Conv2D(64, (3,3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2,2),
    # The fifth convolution
    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 ('horses') and 1 for the other ('humans')
    tf.keras.layers.Dense(1, activation='sigmoid')
])
```

2

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 298, 298, 16)	448
max_pooling2d (MaxPooling2D)	(None, 149, 149, 16)	0
conv2d_1 (Conv2D)	(None, 147, 147, 32)	4640
max_pooling2d_1 (MaxPooling2	(None, 73, 73, 32)	0
conv2d_2 (Conv2D)	(None, 71, 71, 64)	18496
max_pooling2d_2 (MaxPooling2	(None, 35, 35, 64)	0
conv2d_3 (Conv2D)	(None, 33, 33, 64)	36928
max_pooling2d_3 (MaxPooling2	(None, 16, 16, 64)	0
conv2d_4 (Conv2D)	(None, 14, 14, 64)	36928
max_pooling2d_4 (MaxPooling2	(None, 7, 7, 64)	0
flatten (Flatten)	(None, 3136)	0
dense (Dense)	(None, 512)	1606144
dense_1 (Dense)	(None, 1)	513

Total params: 1,704,097
Trainable params: 1,704,097
Non-trainable params: 0

3

```
from tensorflow.keras.optimizers import RMSprop

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

4

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

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

# Flow training images in batches of 128 using train_datagen generator
train_generator = train_datagen.flow_from_directory(
        '/tmp/horse-or-human/',  # This is the source directory for training images
        target_size=(300, 300),  # All images will be resized to 150x150
        batch_size=128,
        # Since we use binary_crossentropy loss, we need binary labels
        class_mode='binary')
```

Found 1027 images belonging to 2 classes.

5

```
history = model.fit(
      train_generator,
      steps_per_epoch=8,
      epochs=15,
      verbose=1)
```

6

```
history = model.fit_generator(
      train_generator,
      steps_per_epoch=8,
      epochs=15,
      validation_data=validation_generator,
      validation_steps=8,
      verbose=2)
```

7

```
classes = model.predict(images, batch_size=10)
print(classes[0])
if classes[0]>0.5:
  print(fn + " is a human")
else:
  print(fn + " is a horse")
```

Using generated or real-world images

As you can see we go from the raw pixels of the images to increasingly abstract and compact representations. The representations downstream start highlighting what the network pays attention to, and they show fewer and fewer features being "activated"; most are set to zero. This is called "sparsity." Representation sparsity is a key feature of deep learning.

These representations carry increasingly less information about the original pixels of the image, but increasingly refined information about the class of the image. You can think of a convnet (or a deep network in general) as an information distillation pipeline.

```
import numpy as np
import random
from tensorflow.keras.preprocessing.image import img_to_array, load_img

# Let's define a new Model that will take an image as input, and will output
# intermediate representations for all layers in the previous model after
# the first.

my_layers = model.layers
#my_layers = model.layers[1:]

successive_outputs = [layer.output for layer in my_layers]
#visualization_model = Model(img_input, successive_outputs)
visualization_model = tf.keras.models.Model(inputs = model.input, outputs = successive_outputs)

user_uploaded_file = True
#user_uploaded_file = False

if user_uploaded_file == False:
  # Let's prepare a random input image from the training set.
  horse_img_files = [os.path.join(train_horse_dir, f) for f in train_horse_names]
  human_img_files = [os.path.join(train_human_dir, f) for f in train_human_names]
  img_path = random.choice(horse_img_files + human_img_files)


  img = load_img(img_path, target_size=(300, 300))  # this is a PIL image
  x = img_to_array(img)  # Numpy array with shape (150, 150, 3)
  x = x.reshape((1,) + x.shape)  # Numpy array with shape (1, 150, 150, 3)

# Rescale by 1/255
x /= 255

classes = model.predict(x)
print(classes[0])
if classes[0]>0.5:
  print("Image is of a human")
else:
  print("Image is of a horse")
```

```
# Let's run our image through our network, thus obtaining all
# intermediate representations for this image.
successive_feature_maps = visualization_model.predict(x)

# These are the names of the layers, so can have them as part of our plot
layer_names = [layer.name for layer in my_layers]

# Now let's display our representations
for layer_name, feature_map in zip(layer_names, successive_feature_maps):
  print(layer_name, feature_map.shape, len(feature_map.shape))
  if len(feature_map.shape) == 4:
    # Just do this for the conv / maxpool layers, not the fully-connected layers, conv2d (1, 149, 149, 16) 4
    n_features = feature_map.shape[-1]  # number of features in feature map
    # The feature map has shape (1, size, size, n_features)
    size = feature_map.shape[1]
    # We will tile our images in this matrix
    display_grid = np.zeros((size, size * n_features))
    for i in range(n_features):
      # Postprocess the feature to make it visually palatable
      x = feature_map[0, :, :, i]
      x -= x.mean()
      x /= x.std()
      x *= 64
      x += 128
      x = np.clip(x, 0, 255).astype('uint8')
      # We'll tile each filter into this big horizontal grid
      display_grid[:, i * size : (i + 1) * size] = x
    # Display the grid
    scale = 20. / n_features
    plt.figure(figsize=(scale * n_features, scale))
    plt.title(layer_name)
    plt.grid(False)
    plt.imshow(display_grid, aspect='auto', cmap='viridis')
```

```
[0.03831019]
Image is of a horse
conv2d (1, 298, 298, 16) 4
max_pooling2d (1, 149, 149, 16) 4
conv2d_1 (1, 147, 147, 32) 4
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:61: RuntimeWarning: invalid value encountered in
max_pooling2d_1 (1, 73, 73, 32) 4
conv2d_2 (1, 71, 71, 64) 4
max_pooling2d_2 (1, 35, 35, 64) 4
conv2d_3 (1, 33, 33, 64) 4
max_pooling2d_3 (1, 16, 16, 64) 4
conv2d_4 (1, 14, 14, 64) 4
max_pooling2d_4 (1, 7, 7, 64) 4
flatten (1, 3136) 2
dense (1, 512) 2
dense_1 (1, 1) 2
```

```
model.layers
```

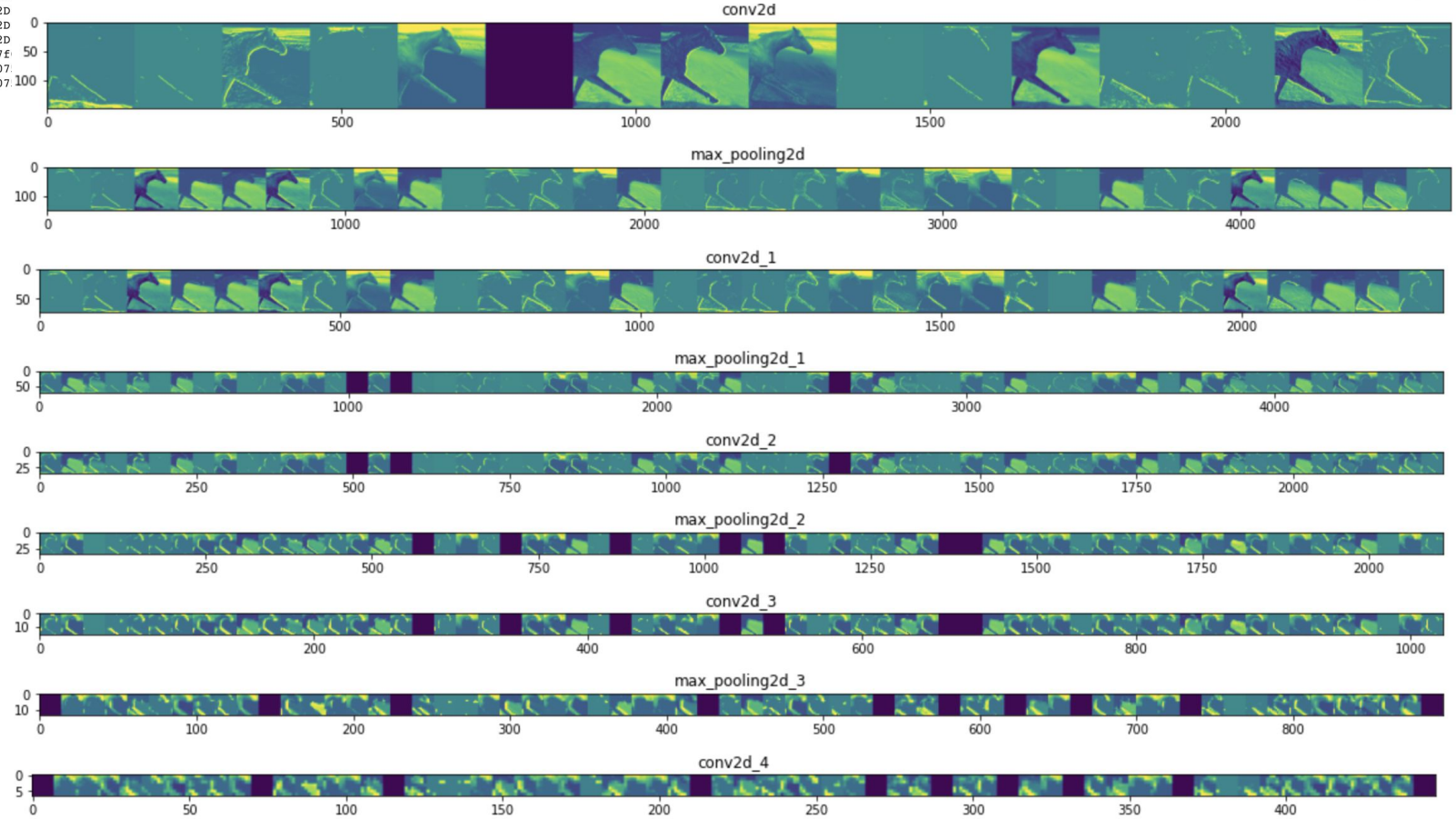
```
convolutional.Conv2D
pooling.MaxPooling2D
convolutional.Conv2D
pooling.MaxPooling2D
convolutional.Conv2D
pooling.MaxPooling2D
convolutional.Conv2D
pooling.MaxPooling2D
convolutional.Conv2D
pooling.MaxPooling2D
core.Flatten at 0x7f
core.Dense at 0x7f07
core.Dense at 0x7f07
```

Using generated or real-world images

```
model.layers[0].Conv2D
```

```
successive_outputs = [layer.output for layer in model.layers[1:]]
```

Layer (type)	Output Shape
conv2d (Conv2D)	(None, 298, 298, 16)
max_pooling2d (MaxPooling2D)	(None, 149, 149, 16)
conv2d_1 (Conv2D)	(None, 147, 147, 32)
max_pooling2d_1 (MaxPooling2	(None, 73, 73, 32)
conv2d_2 (Conv2D)	(None, 71, 71, 64)
	2 (None, 35, 35, 64)
	(None, 33, 33, 64)
	2 (None, 16, 16, 64)
	(None, 14, 14, 64)
	2 (None, 7, 7, 64)
	(None, 3136)
	(None, 512)
	(None, 1)



```
model.layers
```

```
convolutional.Conv2D
pooling.MaxPooling2D
convolutional.Conv2D
pooling.MaxPooling2D
convolutional.Conv2D
pooling.MaxPooling2D
convolutional.Conv2D
pooling.MaxPooling2D
convolutional.Conv2D
pooling.MaxPooling2D
core.Flatten at 0x7f
core.Dense at 0x7f07
core.Dense at 0x7f07
```

```
model.layers[0].Conv2D
```

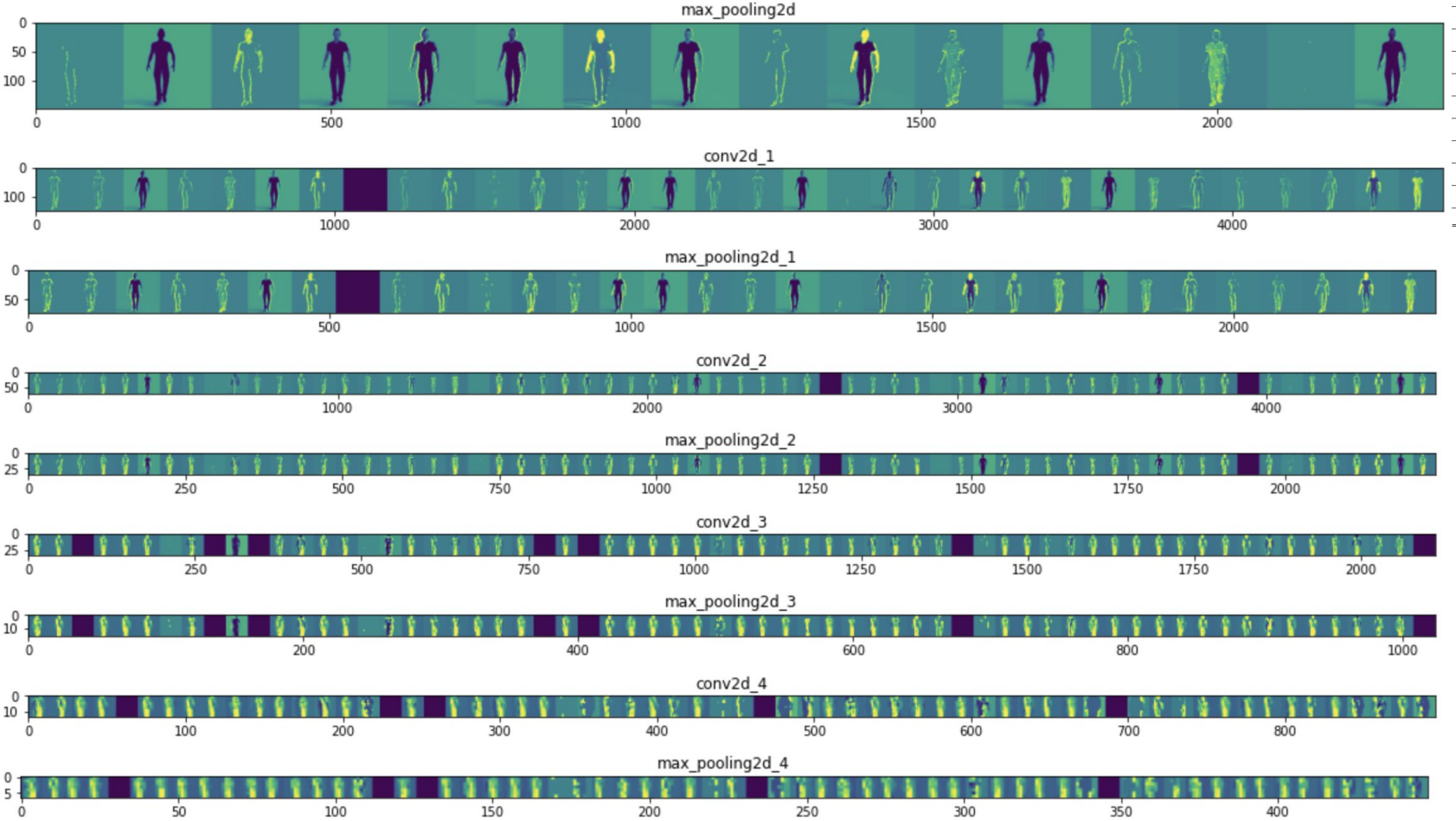
- **valhuman01-00.png**(image/png) -

```
Saving valhuman01-00.png to
[1.]
valhuman01-00.png is a human
```

[1.]

```
successive_outputs = [layer.output for layer in model.layers[1:]]
```

Layer (type)	Output Shape
conv2d (Conv2D)	(None, 298, 298, 16)
max_pooling2d (MaxPooling2D)	(None, 149, 149, 16)
conv2d_1 (Conv2D)	(None, 147, 147, 32)
	73, 73, 32)
	71, 71, 64)
	35, 35, 64)
	33, 33, 64)
	16, 16, 64)
	14, 14, 64)
	7, 7, 64)
	3136)
	512)
	1)



```
model.layers
```

```
convolutional.Conv2D
pooling.MaxPooling2D
convolutional.Conv2D
pooling.MaxPooling2D
convolutional.Conv2D
pooling.MaxPooling2D
convolutional.Conv2D
pooling.MaxPooling2D
convolutional.Conv2D
pooling.MaxPooling2D
core.Flatten at 0x7f
core.Dense at 0x7f07
core.Dense at 0x7f07
```

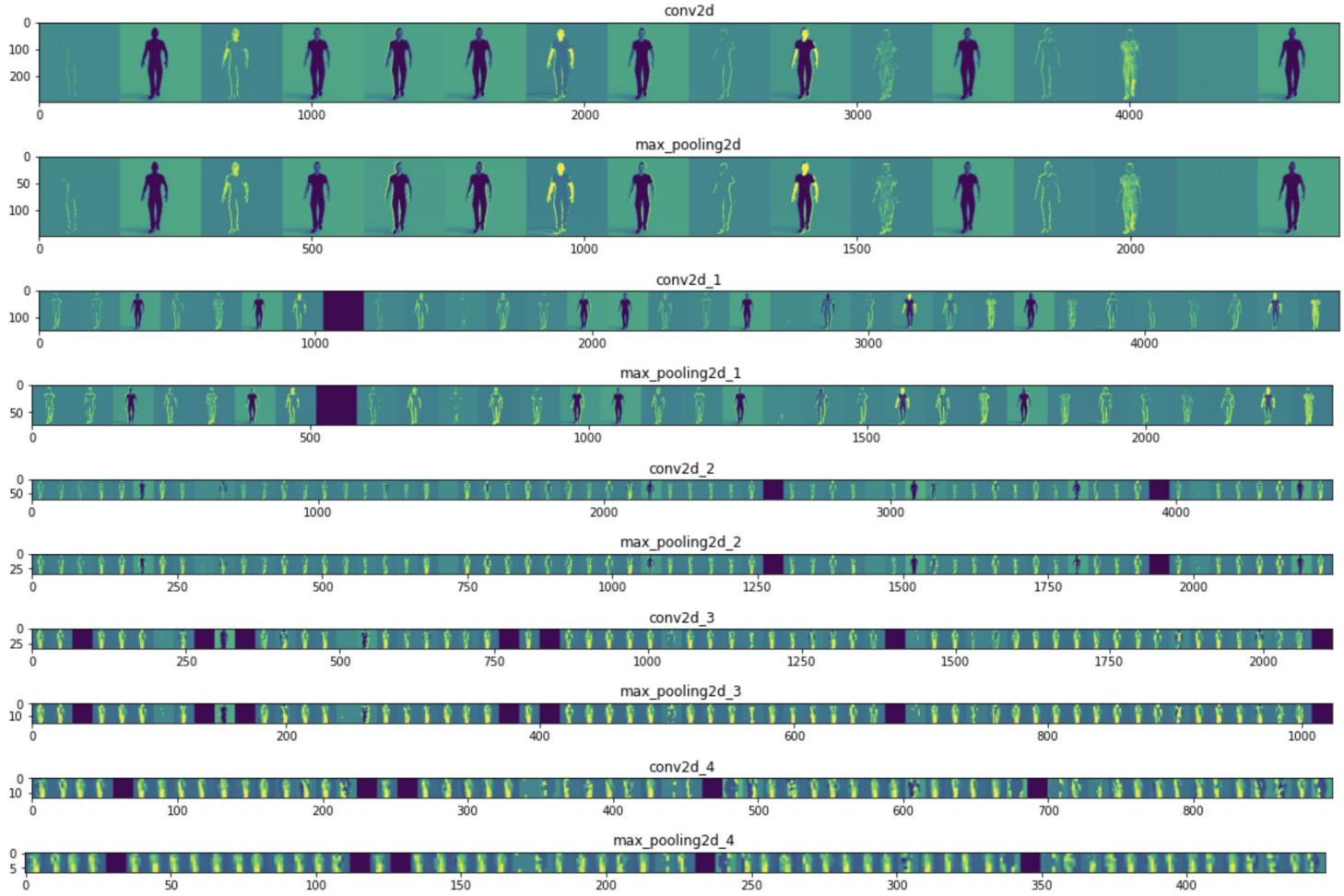
```
model.layers[0].Conv2D
```

- **valhuman01-00.png**(image/png) -

```
Saving valhuman01-00.png to
[1.]
valhuman01-00.png is a human
```

[1.]

```
successive_outputs = [layer.output for layer in model.layers]
```



Layer (type)	Output Shape
conv2d (Conv2D)	(None, 298, 298, 16)
max_pooling2d (MaxPooling2D)	(None, 149, 149, 16)
_1 (Conv2D)	(None, 147, 147, 32)
oling2d_1 (MaxPooling2	(None, 73, 73, 32)
_2 (Conv2D)	(None, 71, 71, 64)
oling2d_2 (MaxPooling2	(None, 35, 35, 64)
_3 (Conv2D)	(None, 33, 33, 64)
oling2d_3 (MaxPooling2	(None, 16, 16, 64)
_4 (Conv2D)	(None, 14, 14, 64)
oling2d_4 (MaxPooling2	(None, 7, 7, 64)
n (Flatten)	(None, 3136)
(Dense)	(None, 512)
1 (Dense)	(None, 1)

```
params: 1,704,097
ble params: 1,704,097
ainable params: 0
```

```
model.layers
```

```
convolutional.Conv2D
pooling.MaxPooling2D
convolutional.Conv2D
pooling.MaxPooling2D
convolutional.Conv2D
pooling.MaxPooling2D
convolutional.Conv2D
pooling.MaxPooling2D
convolutional.Conv2D
pooling.MaxPooling2D
core.Flatten at 0x7f
core.Dense at 0x7f07
core.Dense at 0x7f07
```

```
model.layers[0].Conv2D
```

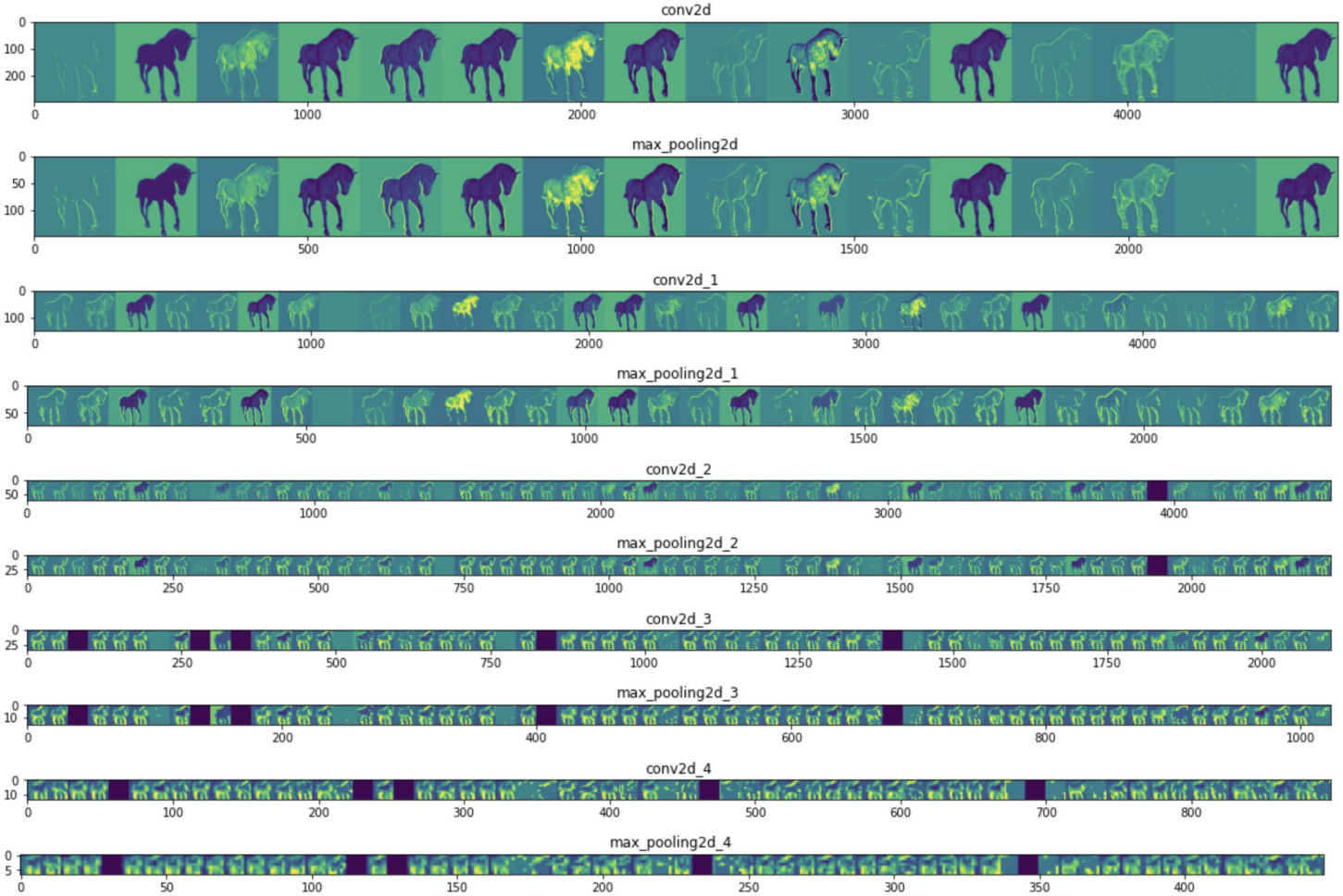
- **horse1-000.png**(image/png) -
Saving horse1-000.png to
[0.]
horse1-000.png is a horse

[6.4249965e-05]

```
successive_outputs = [layer.output for layer in model.layers]
```

Layer (type)	Output Shape
conv2d (Conv2D)	(None, 298, 298, 16)
max_pooling2d (MaxPooling2D)	(None, 149, 149, 16)
l (Conv2D)	(None, 147, 147, 32)
ling2d_1 (MaxPooling2	(None, 73, 73, 32)
2 (Conv2D)	(None, 71, 71, 64)
ling2d_2 (MaxPooling2	(None, 35, 35, 64)
3 (Conv2D)	(None, 33, 33, 64)
ling2d_3 (MaxPooling2	(None, 16, 16, 64)
4 (Conv2D)	(None, 14, 14, 64)
ling2d_4 (MaxPooling2	(None, 7, 7, 64)
(Flatten)	(None, 3136)
Dense)	(None, 512)
(Dense)	(None, 1)

rams: 1,704,097
le params: 1,704,097
inable params: 0



```
model.layers
```

```
convolutional.Conv2D
pooling.MaxPooling2D
convolutional.Conv2D
pooling.MaxPooling2D
convolutional.Conv2D
pooling.MaxPooling2D
convolutional.Conv2D
pooling.MaxPooling2D
convolutional.Conv2D
pooling.MaxPooling2D
core.Flatten at 0x7f
core.Dense at 0x7f07
core.Dense at 0x7f07
```

```
model.layers[0].Conv2D
```

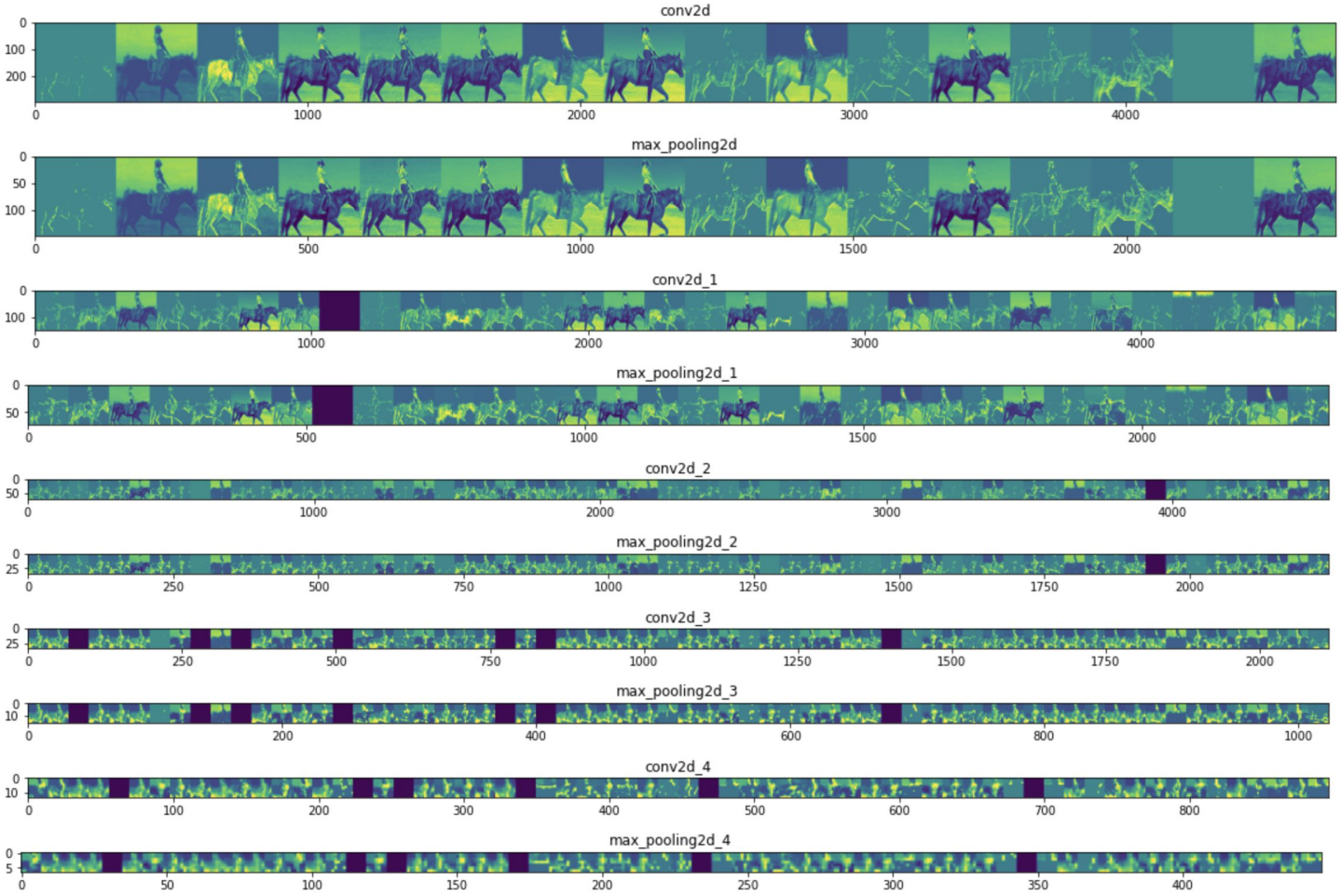
- **human-horse.jpg**(image/jpeg) - Saving human-horse.jpg to h
[0.]
human-horse.jpg is a horse

[0.03831019]

```
successive_outputs = [layer.output for layer in model.layers]
```

Layer (type)	Output Shape
conv2d (Conv2D)	(None, 298, 298, 16)
max_pooling2d (MaxPooling2D)	(None, 149, 149, 16)
conv2d_1 (Conv2D)	(None, 147, 147, 32)
ling2d_1 (MaxPooling2	(None, 73, 73, 32)
_2 (Conv2D)	(None, 71, 71, 64)
ling2d_2 (MaxPooling2	(None, 35, 35, 64)
_3 (Conv2D)	(None, 33, 33, 64)
ling2d_3 (MaxPooling2	(None, 16, 16, 64)
_4 (Conv2D)	(None, 14, 14, 64)
ling2d_4 (MaxPooling2	(None, 7, 7, 64)
(Flatten)	(None, 3136)
Dense)	(None, 512)
(Dense)	(None, 1)

arams: 1,704,097
le params: 1,704,097
inable params: 0



Using Real-world Images

```
=================================================================
conv2d (Conv2D)              (None, 298, 298, 16)      448
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 149, 149, 16)      0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 147, 147, 32)      4640
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 73, 73, 32)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 71, 71, 64)        18496
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 35, 35, 64)        0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 33, 33, 64)        36928
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 16, 16, 64)        0
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 14, 14, 64)        36928
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 7, 7, 64)          0
_________________________________________________________________
flatten (Flatten)            (None, 3136)              0
_________________________________________________________________
dense (Dense)                (None, 512)               1606144
_________________________________________________________________
dense_1 (Dense)              (None, 1)                 513
=================================================================
Total params: 1,704,097
Trainable params: 1,704,097
Non-trainable params: 0
```

```
9s 951ms/step - loss: 0.1559 - acc: 0.9445 - val_loss: 0.9205 - val_acc: 0.8711

2s 210ms/step - loss: 1.3037 - acc: 0.8477
9s 966ms/step - loss: 0.3236 - acc: 0.9114 - val_loss: 1.3037 - val_acc: 0.8477

2s 202ms/step - loss: 1.3805 - acc: 0.8555
8s 937ms/step - loss: 0.0348 - acc: 0.9854 - val_loss: 1.3805 - val_acc: 0.8555

8/8 [==============================] - 2s 209ms/step - loss: 1.4925 - acc: 0.8242
9/9 [==============================] - 9s 968ms/step - loss: 0.0563 - acc: 0.9815 - val_loss: 1.4925 - val_acc: 0.8242
Epoch 13/15
8/8 [==============================] - 2s 207ms/step - loss: 1.4278 - acc: 0.8438
9/9 [==============================] - 9s 957ms/step - loss: 0.0232 - acc: 0.9942 - val_loss: 1.4278 - val_acc: 0.8438
Epoch 14/15
8/8 [==============================] - 2s 212ms/step - loss: 0.3939 - acc: 0.8672
9/9 [==============================] - 9s 977ms/step - loss: 0.1954 - acc: 0.9435 - val_loss: 0.3939 - val_acc: 0.8672
Epoch 15/15
8/8 [==============================] - 2s 211ms/step - loss: 1.3280 - acc: 0.8516
9/9 [==============================] - 9s 969ms/step - loss: 0.0939 - acc: 0.9737 - val_loss: 1.3280 - val_acc: 0.8516
```

Using Real-world Images

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

```
9/9 [==============================] - 7s 810ms/step - loss: 0.0052 - acc: 1.0000 - val_loss: 1.5228 - val_acc: 0.8594
Epoch 11/15
8/8 [==============================] - 1s 179ms/step - loss: 1.1527 - acc: 0.8750
9/9 [==============================] - 7s 812ms/step - loss: 0.0033 - acc: 1.0000 - val_loss: 1.1527 - val_acc: 0.8750
Epoch 12/15
2/9 [=====>........................] - ETA: 2s - loss: 0.5746 - acc: 0.8789

Epoch 14/15
8/8 [==============================] - 1s 179ms/step - loss: 2.0000 - acc: 0.8203
9/9 [==============================] - 7s 799ms/step - loss: 0.0027 - acc: 1.0000 - val_loss: 2.0000 - val_acc: 0.8203
Epoch 15/15
8/8 [==============================] - 1s 176ms/step - loss: 0.7265 - acc: 0.8398
9/9 [==============================] - 7s 786ms/step - loss: 0.0069 - acc: 0.9990 - val_loss: 0.7265 - val_acc: 0.8398
```

Course 1: Introduction to TensorFlow for Artificial Intelligence, Machine Learning, and Deep Learning

Week 1: A New Programming Paradigm (see Meetup #1)
Week 2: Introduction to Computer Vision
Week 3: Enhancing Vision with Convolutional Neural Networks
Week 4: Using Real-world Images

Course 1: Introduction to TensorFlow for Artificial Intelligence, Machine Learning, and Deep Learning Questions

Remember the batch size you used when you created it, it was 20, that's important in the next step. There are 1,024 images in the training directory, so we're loading them in 128 at a time. So in order to load them all, we need to do 8 batches. So we set the steps_per_epoch to cover that.

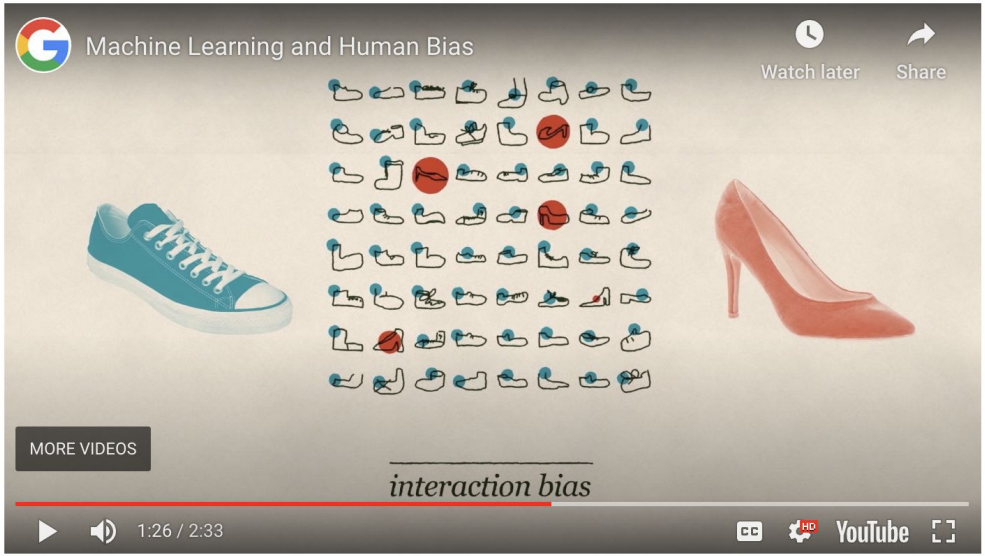
```
train_datagen = ImageDataGenerator(rescale=1./255)

train_generator = train_datagen.flow_from_directory(
        train_dir,
        target_size=(300, 300),
        batch_size=128,
        class_mode='binary')
```

1. %%javascript
2. C1W2: Dense default activation function is linear.
   Flatten() default input shape ? Affect Output shape "multiple"
   tf.keras.layers.Flatten(input_shape=(28,28)
3. (11,11,64) (5,5,64) floored division
4. for x in range(1,size_x-1)?
   size_x is 512
5. CLASS callbacks methods? acc vs loss in callback?
6. tensorboard not showing graph in colab
7. Per Layer execution time? complex layers? plt.colorbar(plt.pcolor(f1[0, : , :, CONVOLUTION_NUMBER], vmin=0, vmax=1))
8. Applying Convolutions on top of our Deep neural network will make training: It depends on many factors ?
9. simplest model to classify data?
10. Special edition on OLM: model.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['accuracy'])
11. https://www.tensorflow.org/api_docs/python/tf/keras/Model steps_per_epoch validation_steps
12. successive_outputs = [layer.output for layer in model.layers[1:]] layer_names = [layer.name for layer in model.layers]

Check out these resources

Machine Learning Fairness

As an AI-first company, Google aims to develop the benefits of machine learning for everyone.

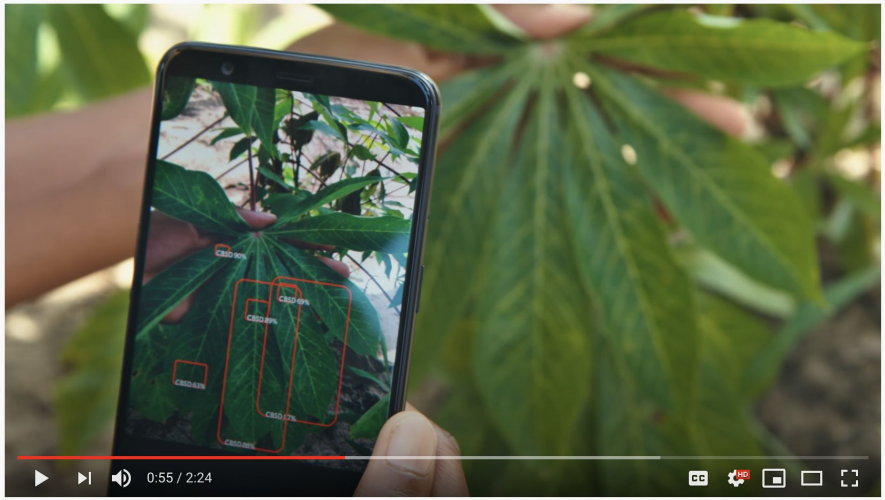


Source: https://developers.google.com/machine-learning/fairness-overview/

Check out these resources

TensorFlow: an ML platform for solving impactful and challenging problems

105,225 views • Mar 30, 2018 2.3K 25 SHARE SAVE



TensorFlow: an ML platform for solving impactful and challenging problems

105,225 views • Mar 30, 2018 2.3K 25 SHARE SAVE

Source: https://www.youtube.com/watch?v=NIpS-DhayQA
https://plantvillage.psu.edu/
https://www.iita.org/
https://www.blog.google/technology/ai/ai-takes-root-helping-farmers-identity-diseased-plants/

Check out these events and Meetups

New AI For Medicine Specialization coming soon!

Thursday, April 9, 2020

TensorFlow Dev Summit Extended Coimbatore 2020

Friday, April 17, 2020

[VIRTUAL MEETUP] Recommender Systems and Deep Models at Etsy

Saturday, April 25, 2020

AI Meet & Greet

Hosted by
viraf and 2 others

DC Natural Language Processing (DC-NLP) + online

Monday, April 20, 2020

Online Python for Data Science Immersive

Saturday, April 25, 2020

EzAnalytix-An Innovative Healthcare Analytic Tool-Presented by Larry Liu

Tuesday, April 28, 2020

Facebook: The Inside Story (Online Book Club)

Thinkful DC | Careers in Tech

Wednesday, April 22, 2020

[VIRTUAL] The Science Behind Netflix Recommendations

Thursday, April 16, 2020

Virtual: Intro to Machine Learning in Cybersecurity

Thursday, April 30, 2020

[ONLINE] Machine Learning Visualization with Yellowbrick

Source:
https://www.youtube.com/watch?v=zGFKSQlef_0
https://www.meetup.com/TFUGCbe/events/269893545/
https://www.meetup.com/Analytics-Data-Science-by-Dataiku-WashingtonDC/events/269958244/
https://www.meetup.com/Bethesda-Artificial-Intelligence-Meetup/events/268561559/
https://www.meetup.com/DC-NLP/
https://www.meetup.com/Python-for-Data-Science-and-Machine-Learning-Washington/events/269525914/
https://www.meetup.com/Health-Economics-and-Outcomes-Research-HEOR/events/269191229/
https://www.meetup.com/CodeColloquy-A-Book-Club-About-Technology-and-Society/events/269567095/
https://www.meetup.com/Thinkful-DC/
https://www.meetup.com/Flatiron-School-DC-Coding-Community/events/269699342/
https://www.meetup.com/Data-Education-DC/events/269836881/
https://www.meetup.com/Statistical-Seminars-DC/events/269573641/

Check out this new certification and online conference

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.

If you're attending a GTC Digital 2 hour instructor-led training session, you can review the session information below.

- Medical Image Classification Using the MedNIST Dataset
- Accelerated Machine Learning at Scale with NVIDIA RAPIDS on Microsoft Azure
- Introduction to CUDA Python with Numba
- Accelerating Data Science Workflows with RAPIDS V2
- Integrating Custom Sensors with DriveWorks CORE
- Optimization and Deployment of TensorFlow Models with TensorRT
- Deployment for Intelligent Video Analytics using TensorRT
- Coarse-to-Fine Contextual Memory for Medical Imaging
- 3-D Segmentation for Medical Imaging with V-Net
- Deep Autoencoders for Recommendation Systems
- Deep Learning at Scale with Horovod
- Anomaly Detection with Variational Autoencoders
- High Performance Computing with Containers
- Data Augmentation and Segmentation with Generative Networks for Medical Imaging
- How to Train Your Conversational AI to Pay Attention: A Hands-On Approach to NLP and Transformers

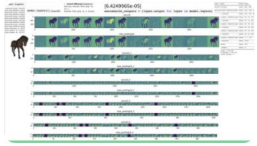
Source:
https://blog.tensorflow.org/2020/03/introducing-tensorflow-developer-certificate.html?m=1
https://www.nvidia.com/en-us/gtc/
https://developer.nvidia.com/dli/getready

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

FRI, APR 24, 7:30 PM EDT

TensorFlow in Practice - Course 1 - Week 4

Online event



Join us for our 3rd adventure in Deep Learning! Just bring your curiosity and be ready to meet our growing community 😀 We are taking Course 1 of TensorFlow in Practice Specialization available at: https://www.coursera.org/specializations/tensorflow-in-practice Join Zoom Meeting: See...

61 attendees 1 Attend

FRI, MAY 1, 7:30 PM EDT

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

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: https://www.coursera.org/specializations/tensorflow-in-practice Join Zoom Meeting: See...

9 attendees Attend

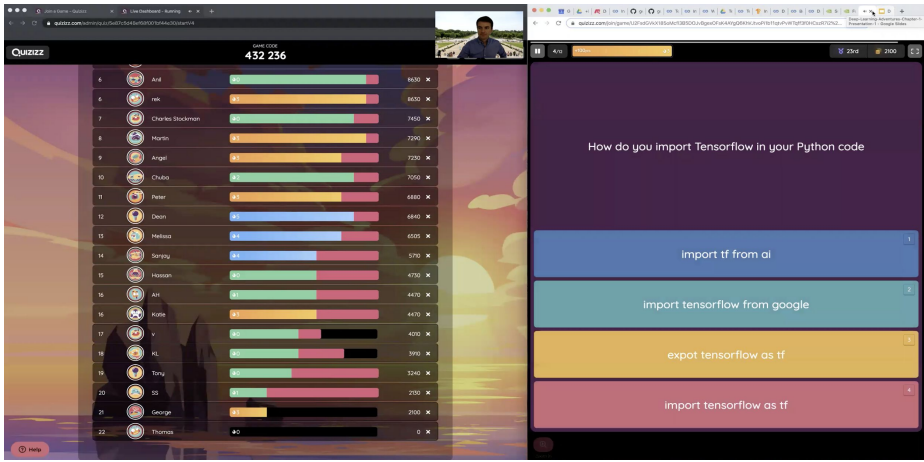
FRI, MAY 8, 7:30 PM EDT

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

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: https://www.coursera.org/specializations/tensorflow-in-practice Join Zoom Meeting: See...

8 attendees Attend



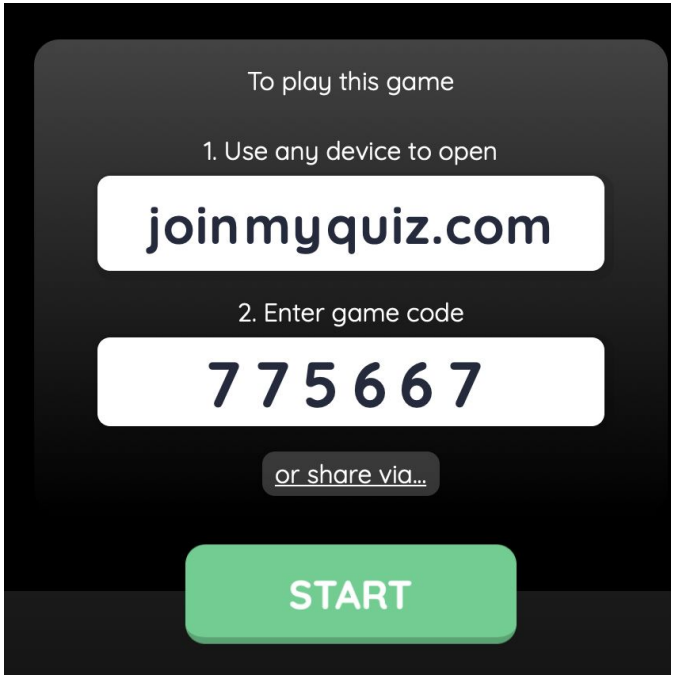
Source: https://www.meetup.com/Deep-Learning-Adventures

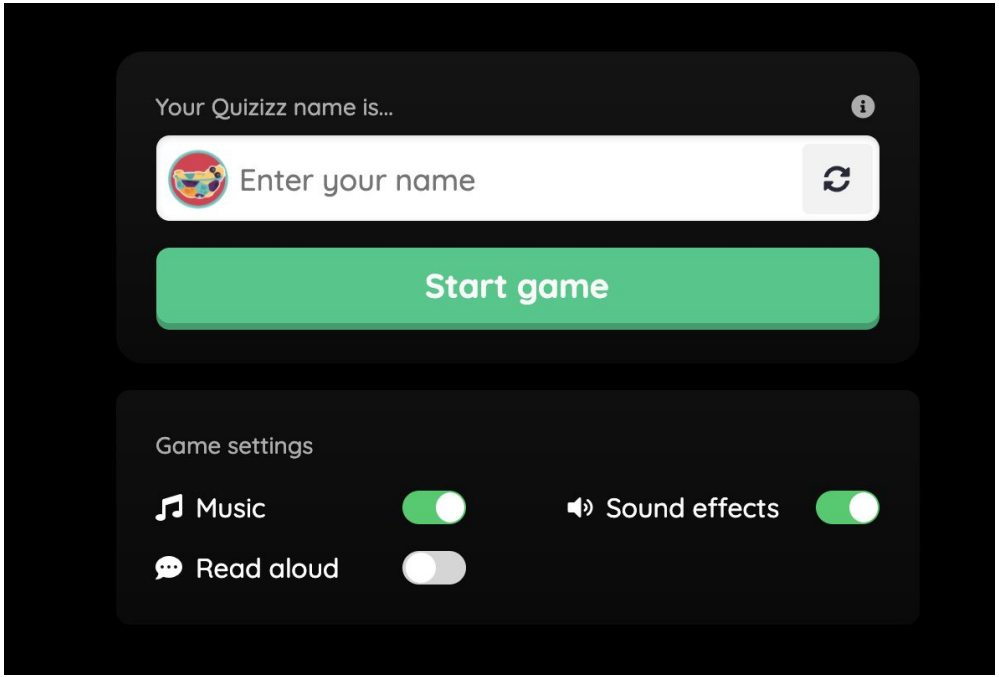
Time for a fun game 😀🎉

Deep Learning Adventures - Quiz 2

Professional Development · 3 times

Other

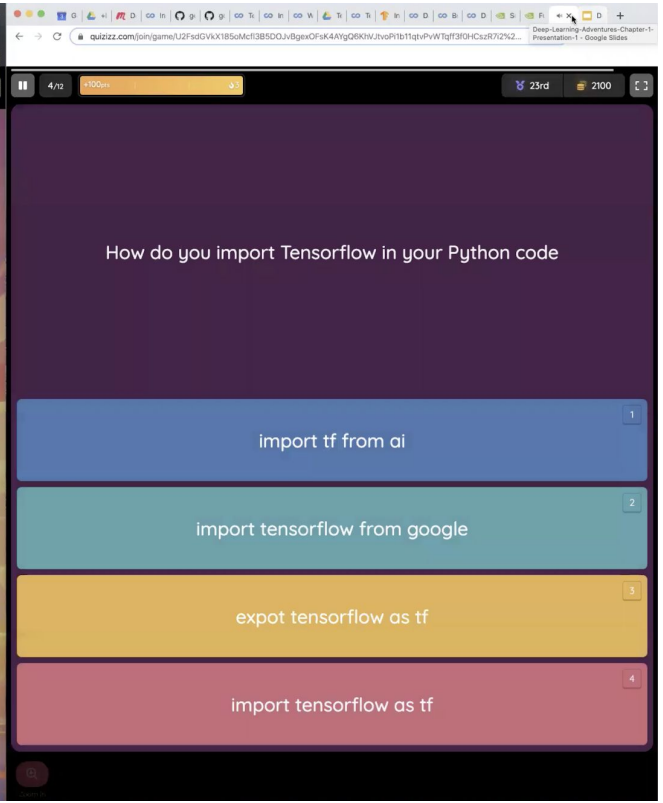




Practice here or use Flashcards:
https://quizizz.com/join/quiz/5e90e8e631fb32001f63510b/start?from=soloLinkShare&referrer=5d921444d0fa99001a135336

Time for a fun game 😀🎉

Practice here or use Flashcards:
quizizz.com/join/quiz/5e87bdbc07fa7f001b120404/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

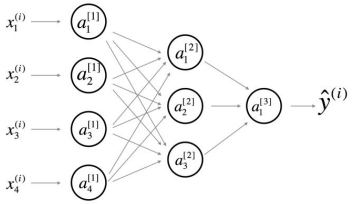


Figure 1: Comprehensive Network: representation commonly used for Neural Networks. For better aesthetic, we omitted the details on the parameters ($w_{ij}^{[l]}$ and $b_i^{[l]}$ etc...) that should appear on the edges

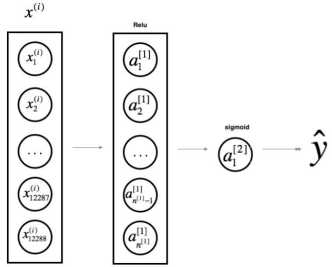


Figure 2: Simplified Network: a simpler representation of a two layer neural network, both are equivalent.

Content added after our 2nd session, based on group feedback

4.3 Normalizing the Inputs

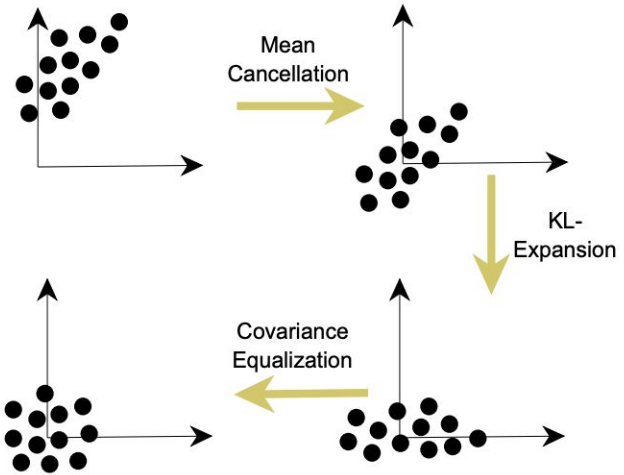
Convergence is usually faster if the average of each input variable over the training set is close to zero. To see this, consider the extreme case where all the inputs are positive. Weights to a particular node in the first weight layer are updated by an amount proportional to δx where δ is the (scalar) error at that node and x is the input vector (see equations (5) and (10)). When all of the components of an input vector are positive, all of the updates of weights that feed into a node will be the same sign (i.e. sign(δ)). As a result, these weights can only all decrease or all increase *together* for a given input pattern. Thus, if a weight vector must change direction it can only do so by zigzagging which is inefficient and thus very slow.

Transforming the Inputs

1. The average of each input variable over the training set should be close to zero.
2. Scale input variables so that their covariances are about the same.
3. Input variables should be uncorrelated if possible.

Yann LeCun
@ylecun

Professor at NYU. Chief AI Scientist at Facebook.
Researcher in AI, Machine Learning, etc.
ACM Turing Award Laureate.

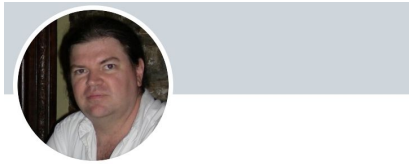


Source: http://yann.lecun.com/exdb/publis/pdf/lecun-98b.pdf
https://www.quora.com/Why-are-image-pixel-values-zero-centered-by-dividing-by-255-before-passing-them-as-input-to-feedforward-neural-network-or-convolutional-NN

Efficient BackProp

Content added after our 2nd session

Abstract. The convergence of back-propagation learning is analyzed so as to explain common phenomenon observed by practitioners. Many undesirable behaviors of backprop can be avoided with tricks that are rarely exposed in serious technical publications. This paper gives some of those tricks, and offers explanations of why they work.
Many authors have suggested that second-order optimization methods are advantageous for neural net training. It is shown that most "classical" second-order methods are impractical for large neural networks. A few methods are proposed that do not have these limitations.



Yann LeCun
@ylecun

Professor at NYU. Chief AI Scientist at Facebook.
Researcher in AI, Machine Learning, etc.
ACM Turing Award Laureate.

4.1 Stochastic versus Batch learning.

4.2 Shuffling the Examples

4.3 Normalizing the Inputs

4.4 The Sigmoid

4.5 Choosing Target Values

4.6 Initializing the weights

4.7 Choosing Learning rates

4.8 Radial Basis Functions vs Sigmoid Units

Source: http://yann.lecun.com/exdb/publis/pdf/lecun-98b.pdf