

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

Start a new group Log in Sign up

The screenshot shows the homepage of the Deep Learning Adventures Meetup group. At the top, there are two photos: a selfie of two people and a generated image of a landscape. Below these are two more generated images labeled "Generated image: 250 iterations" and "Generated image: 1000 iterations". A navigation bar includes links for About, Events, Members, Photos, Discussions, and More. A red "Join this group" button is prominently displayed. The "About" section contains a detailed description of the group's mission and history, followed by a "Read more" link. The "Upcoming events (1)" section lists an event titled "Deep Learning Adventures - Chapter 1" on Friday, April 3, at 7:30 PM, described as an online event. The event description encourages curiosity and introduces the agenda. The "Organizer" section shows a profile picture of George Z. and a "Message" button. The "Members (160)" section shows a grid of member profiles. A "See all" link is visible above the grid.

Deep Learning Adventures

Washington, DC 160 members · Public group Organized by George Z.

Share: [Facebook](#) [Twitter](#) [LinkedIn](#)

About Events Members Photos Discussions More

Join this group

Organizer

George Z. Message

Members (160)

FRI, APR 3, 7:30 PM

Deep Learning Adventures - Chapter 1

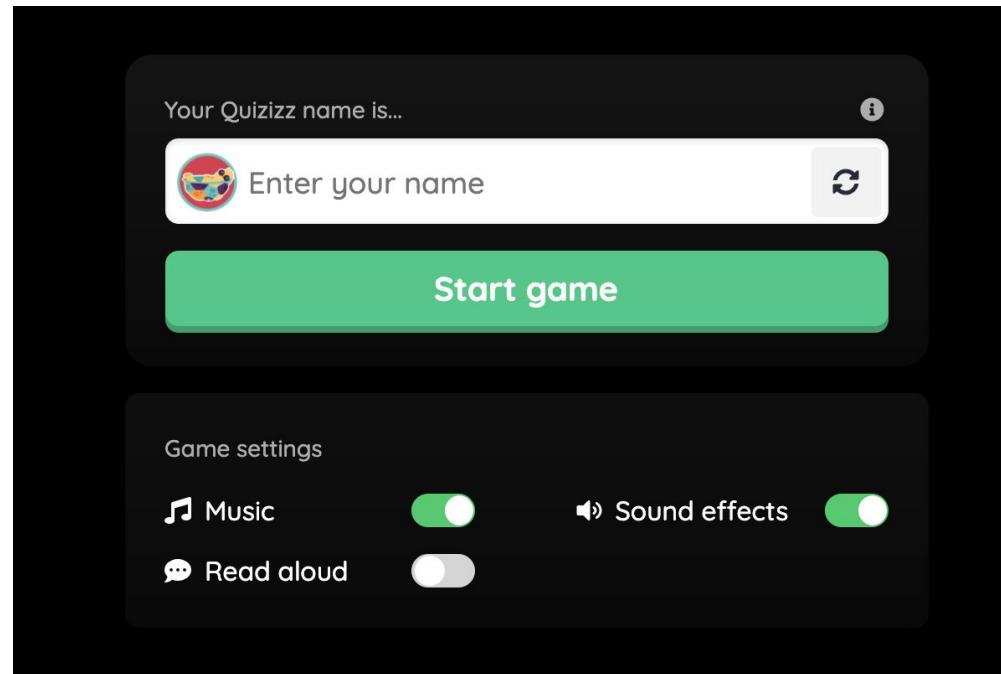
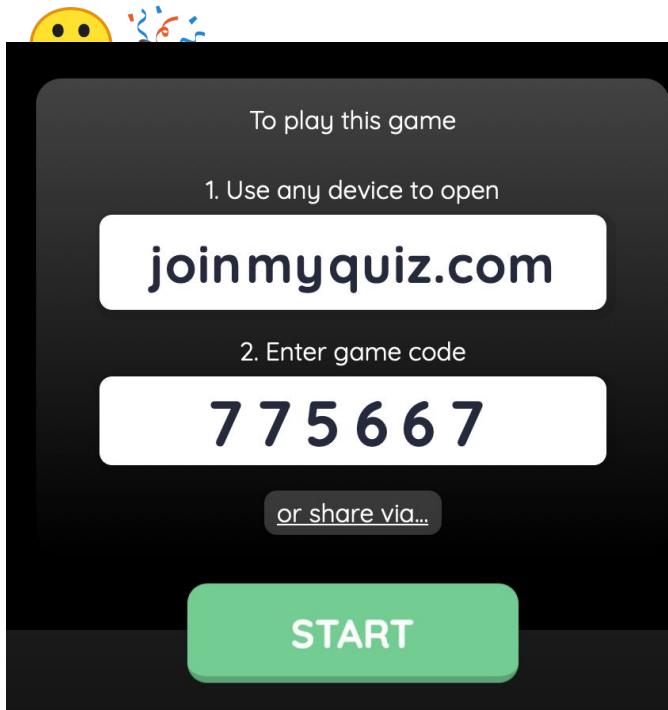
Online event

Join us for our very first adventure in Deep Learning! Just bring your curiosity with you and be ready to meet our growing community 😊 Agenda: - Introductions - What is this cool meetup about? - Industry updates and other interesting meetups...

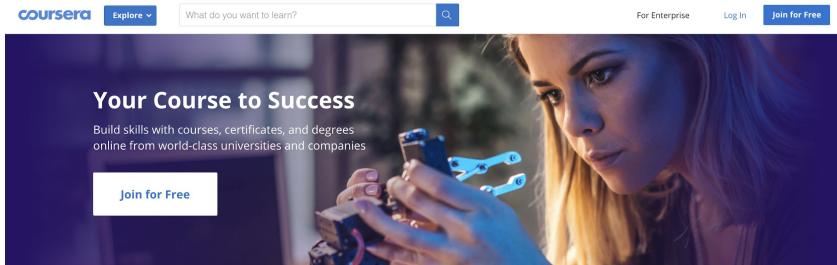
- Meetup Link
<https://www.meetup.com/Deep-Learning-Adventures>
- GitHub repository
<https://github.com/georgezoto/Deep-Learning-Adventures>
- Join us on Slack
https://join.slack.com/t/deeplearninga-nmk8930/shared_invite/zt-d52h9mm9-h~Q0ZXw5PXsTDzPlINlvog

- YouTube recordings of all Meetups
- https://www.youtube.com/playlist?list=PL2EBNWEnwV2dlziH1cJU_RyWipsoYHzN2p

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



Attribution to Coursera and deeplearning.ai



A screenshot of the deeplearning.ai website homepage. The header features the deeplearning.ai logo and a navigation menu with links for "Courses", "Workers", "The Batch", "Events", "Forums", "Blog", and "Company". The main section has a dark background with the title "Break Into AI" in large white letters. Below the title, a paragraph reads: "Whether you want to build algorithms or build a company, deeplearning.ai's courses will teach you key concepts and applications of AI." A red button at the bottom says "Take the Deep Learning Specialization". To the right, there's a graphic of a laptop screen showing two images: a "Content Image" of the Golden Gate Bridge and a "Generated Image" of the same bridge in a painterly style, labeled "Art Generation with Deep Learning".

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](#).

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

★★★★★ 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](#)



6 hours to complete

A New Programming Paradigm

Welcome to this course on going from Basics to Mastery of TensorFlow. We're excited you're here! In week 1 you'll get a soft introduction to what Machine Learning and Deep Learning are, and how they offer you a new programming paradigm, giving you a new set of tools to open previously unexplored scenarios. All you need to know is some very basic SHOW ALL



4 videos (Total 16 min), 5 readings, 3 quizzes [SEE ALL](#)

COURSE

2

Convolutional Neural Networks in TensorFlow

★★★★★ 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](#)



7 hours to complete

Introduction to Computer Vision

Welcome to week 2 of the course! In week 1 you learned all about how Machine Learning and Deep Learning is a new programming paradigm. This week you're going to take that to the next level by beginning to solve problems of computer vision with just a few lines of code! SHOW ALL



7 videos (Total 15 min), 6 readings, 3 quizzes [SEE ALL](#)

COURSE

3

Natural Language Processing in TensorFlow

★★★★★ 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](#)



8 hours to complete

Enhancing Vision with Convolutional Neural Networks

Welcome to week 3! In week 2 you saw a basic Neural Network for Computer Vision. It did the job nicely, but it was a little naive in its approach. This week we'll see how to make it better, as discussed by Laurence and Andrew here. SHOW ALL



6 videos (Total 19 min), 6 readings, 3 quizzes [SEE ALL](#)

COURSE

4

Sequences, Time Series and Prediction

★★★★★ 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](#)



9 hours to complete

Using Real-world Images

Last week you saw how to improve the results from your deep neural network using convolutions. It was a good start, but the data you used was very basic. What happens when your images are larger, or if the features aren't always in the same place? Andrew and Laurence discuss this to prepare you for what you'll learn this week: handling complex images! 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

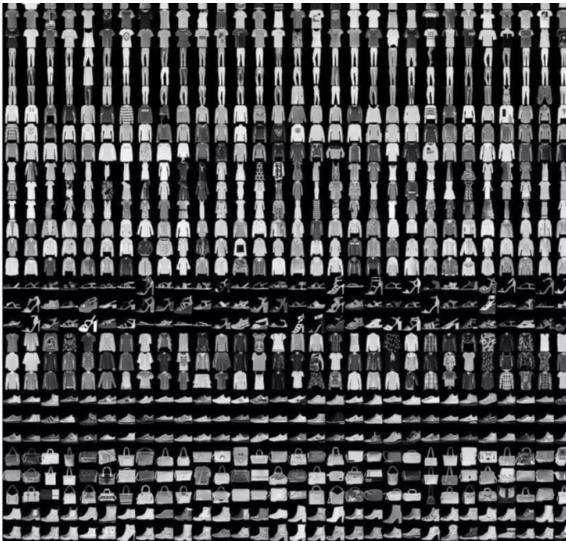
Week 3: Enhancing Vision with Convolutional Neural Networks

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!



deeplearning.ai

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

[zalandoresearch / fashion-mnist](https://github.com/zalandoresearch/fashion-mnist)

Watch 289

Unstar 7.5k

Fork 1.7k

Code

Issues 19

Pull requests 2

Actions

Security

Insights

A MNIST-like fashion product database. Benchmark  <http://fashion-mnist.s3-website.eu-central-1.amazonaws.com/>

[mnist](#) [deep-learning](#) [benchmark](#) [machine-learning](#) [dataset](#) [computer-vision](#) [fashion](#) [fashion-mnist](#) [gan](#) [zalando](#)
[convolutional-neural-networks](#)

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



deeplearning.ai

Visualization: t-SNE

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



An Introduction to computer vision

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

mnist = tf.keras.datasets.fashion_mnist

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

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)
```

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*.

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

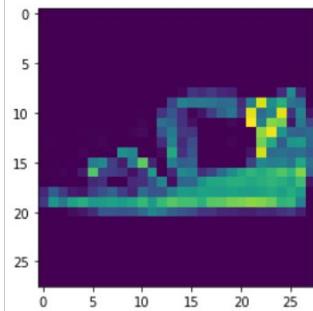
Epoch 1/5
1875/1875 [=====] - 4s 2ms/step - loss: 0.4913 - accuracy: 0.8274
Epoch 2/5
1875/1875 [=====] - 4s 2ms/step - loss: 0.3750 - accuracy: 0.8650
Epoch 3/5
1875/1875 [=====] - 4s 2ms/step - loss: 0.3350 - accuracy: 0.8772
Epoch 4/5
1875/1875 [=====] - 4s 2ms/step - loss: 0.3139 - accuracy: 0.8855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

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/

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])
```



```
training_history = model.fit
```

```
np.average(training_history.history['loss']))  
: ", np.average(training_history.history['accuracy']))  
  
Average training loss:  0.3616836369037628  
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 #  
=====br/>flatten_4 (Flatten)         (None, 784)                0  
  
dense_8 (Dense)             (None, 1024)              803840  
  
dense_9 (Dense)             (None, 10)                 10250  
=====br/>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()
```

```
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>
```

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

An Introduction to computer vision

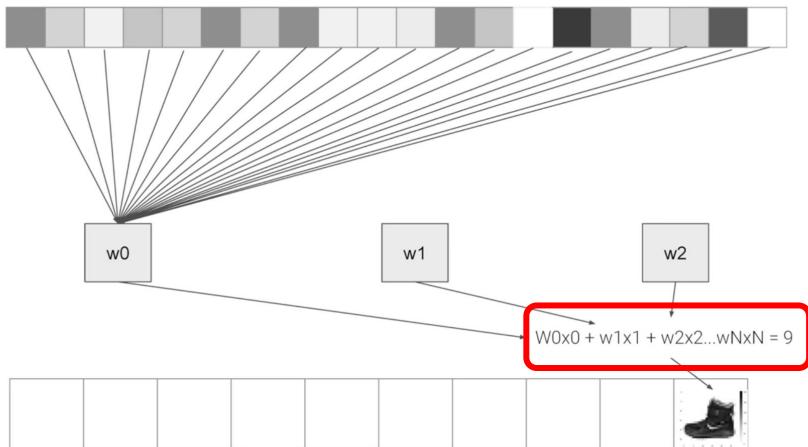
```
import tensorflow as tf
print(tf.__version__)

mnist = tf.keras.datasets.mnist

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

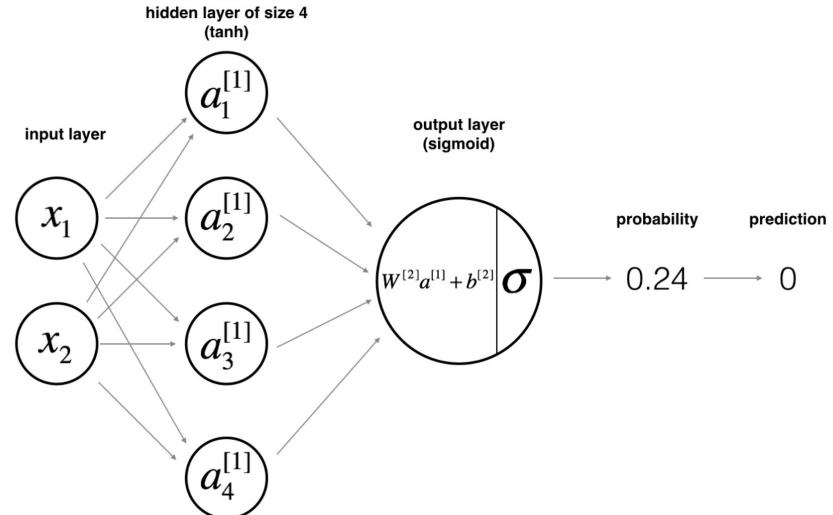
training_images = training_images/255.0
test_images = test_images/255.0

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)])
```



Sample architecture of a 2 layer network (1 hidden + 1 output layer)

It has 2 inputs/features (x_1 and x_2), 1 hidden layer of 4 nodes/neurons (a_1 through a_4) and final output layer of 1 node/neuron



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

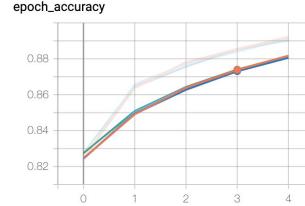
An Introduction to computer vision

```
# 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])
```

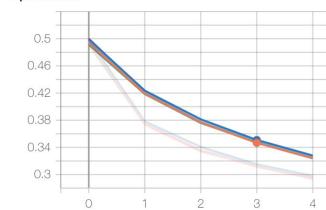
epoch_accuracy



epoch_loss



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

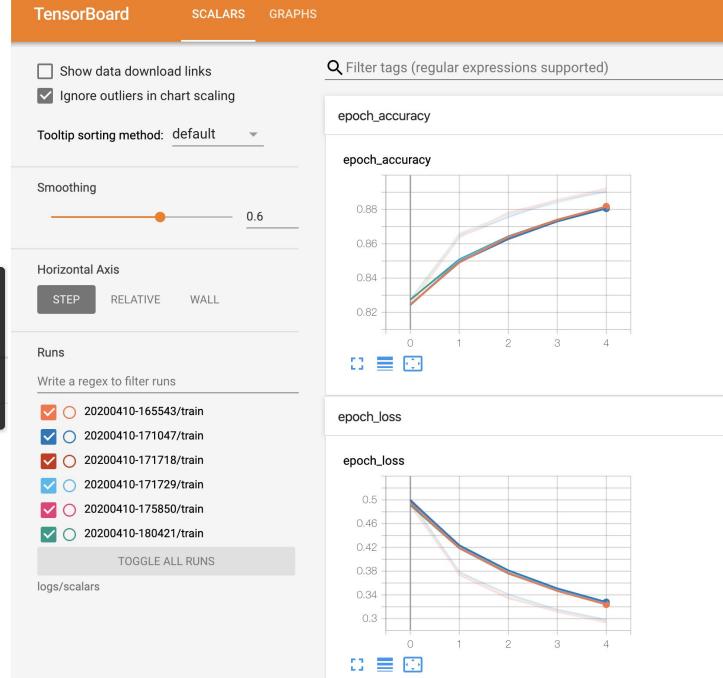


Op-level graph

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.)



TensorBoard

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

Laurence's GitHub repository

<https://github.com/lmoroney/dlaicourse>

Watch 181 ⚡ Star 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

File	Commit Message	Time Ago
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	nbfmft	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	nbfmft	6 days ago
Course 1 - Part 6 - Lesson 3 - Notebook.ipynb	nbfmft	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.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 (Cats v Dogs Augmentation).ipynb	acc => accuracy	6 days ago
Course 2 - Part 4 - Lesson 4 - Notebook.ipynb	acc => accuracy	6 days ago
Hello_World_Layers.ipynb	nbfmft	6 days ago
Horse_or_Human_WithDropouts.ipynb	acc => accuracy	6 days ago
Horse_or_Human_NoValidation.ipynb	acc => accuracy	6 days ago

<https://github.com/lmoroney/dlaicourse>

Watch 183 ⚡ Star 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 nbfmft b8d7f9e 8 days ago

4 contributors

285 lines (285 sloc) | 10.2 KB

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

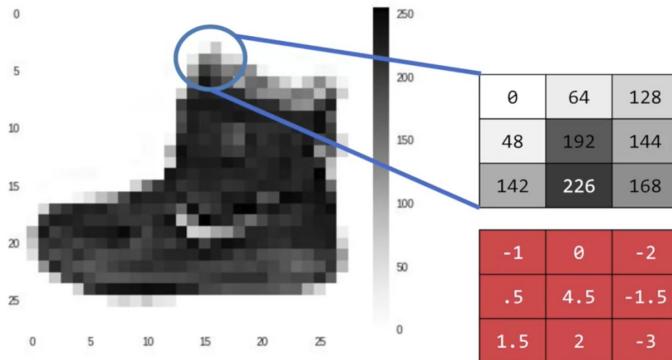
Week 1: A New Programming Paradigm (see Meetup #1)

Week 2: Introduction to Computer Vision

Week 3: Enhancing Vision with Convolutional Neural Networks

Week 4: Using Real-world Images

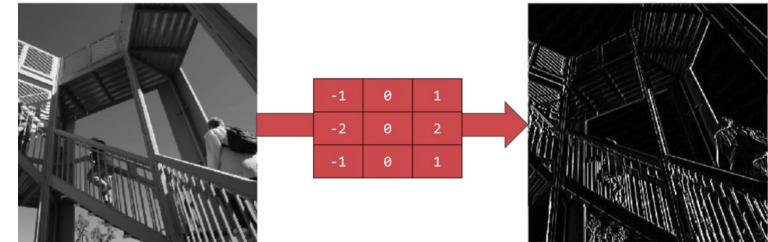
Enhancing Vision with Convolutional Neural Networks



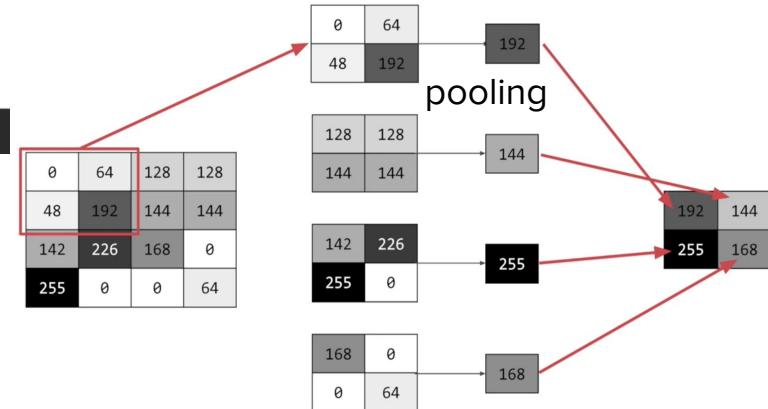
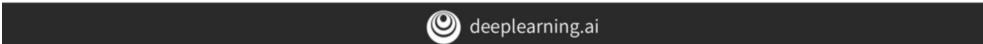
Current Pixel Value is 192

Consider neighbor Values

Filter Definition



$$\begin{aligned} \text{CURRENT_PIXEL_VALUE} &= 192 \\ \text{NEW_PIXEL_VALUE} &= (-1 * 0) + (0 * 64) + (-2 * 128) + \\ &\quad (.5 * 48) + (4.5 * 192) + (-1.5 * 144) + \\ &\quad (1.5 * 42) + (2 * 226) + (-3 * 168) \end{aligned}$$



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

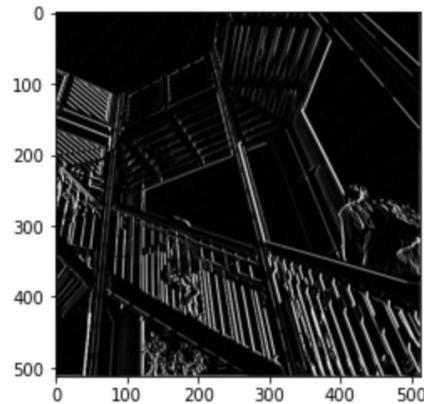
Enhancing Vision with Convolutional Neural Networks

```
# 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
```

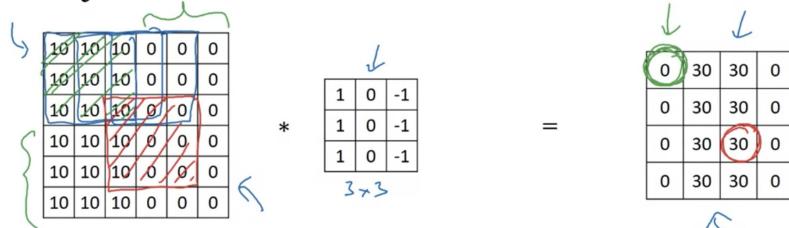


```
for x in range(1,size_x-1):
    for y in range(1,size_y-1):
        convolution = 0.0
        convolution = convolution + (i[x - 1, y-1] * filter[0][0])
        convolution = convolution + (i[x, y-1] * filter[0][1])
        convolution = convolution + (i[x + 1, y-1] * filter[0][2])
        convolution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



Enhancing Vision with Convolutional Neural Networks

Why convolutions

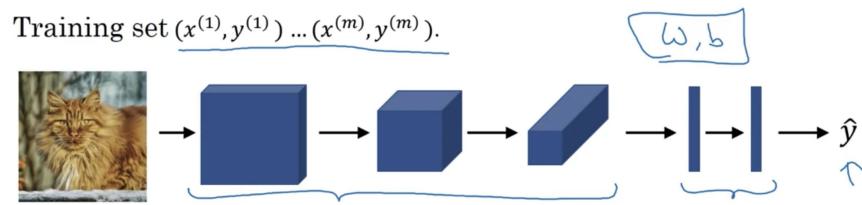


Parameter sharing: A feature detector (such as a vertical edge detector) that's useful in one part of the image is probably useful in another part of the image.

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

Putting it together

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



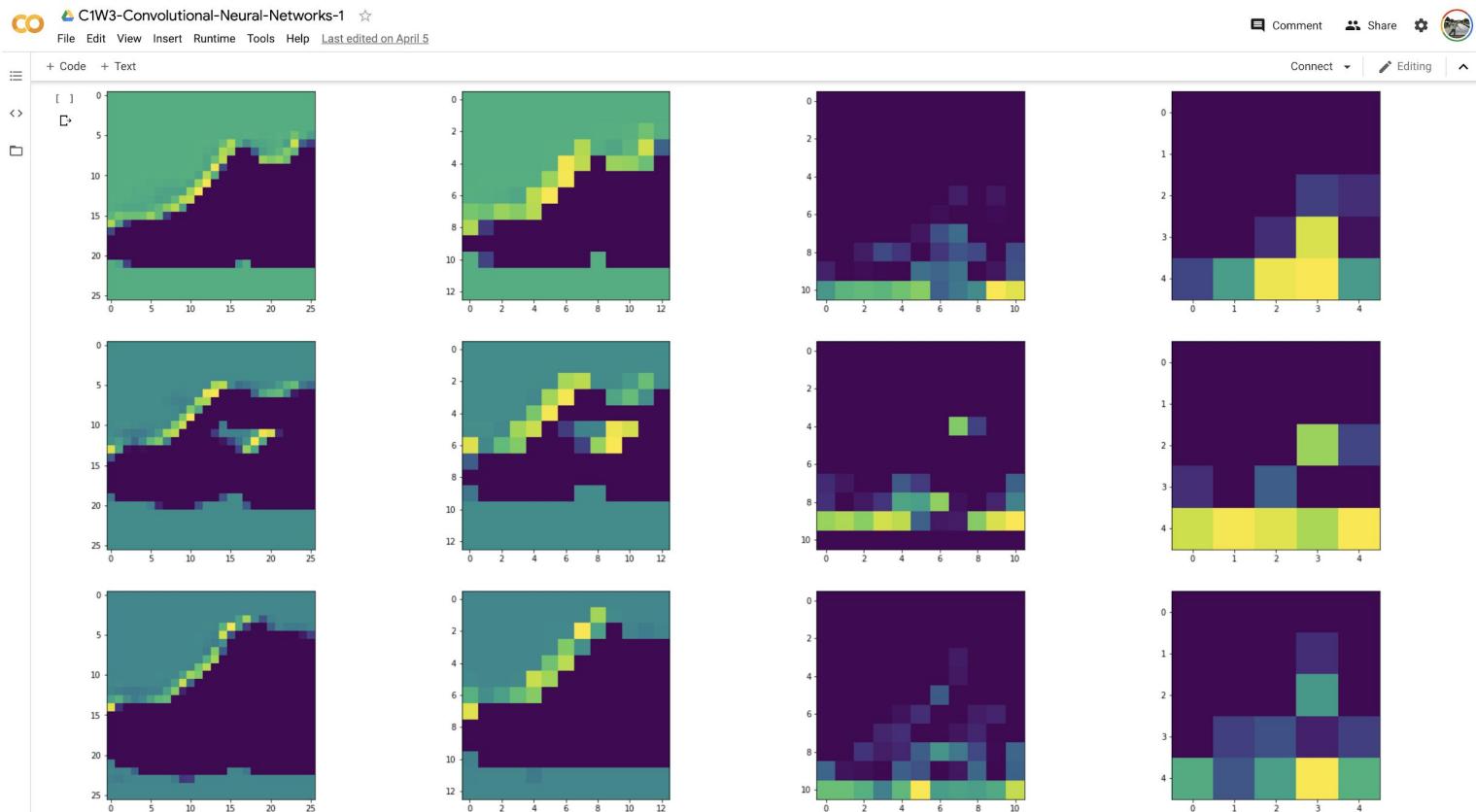
$$\text{Cost } J = \frac{1}{m} \sum_{i=1}^m \mathcal{L}(\hat{y}^{(i)}, y^{(i)})$$

Use gradient descent to optimize parameters to reduce J

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

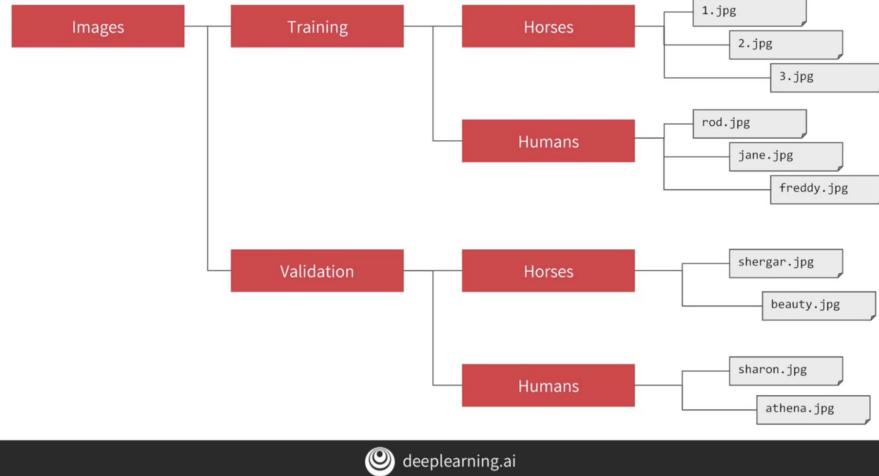
Week 1: A New Programming Paradigm (see Meetup #1)

Week 2: Introduction to Computer Vision

Week 3: Enhancing Vision with Convolutional Neural Networks

Week 4: Using Real-world Images

Using Real-world Images



deeplearning.ai

```
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')
```

1 Using Real-world Images

```
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')
])
```

```
from tensorflow.keras.optimizers import RMSprop

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

3

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 (MaxPooling2D)	(None, 73, 73, 32)	0
conv2d_2 (Conv2D)	(None, 71, 71, 64)	18496
max_pooling2d_2 (MaxPooling2D)	(None, 35, 35, 64)	0
conv2d_3 (Conv2D)	(None, 33, 33, 64)	36928
max_pooling2d_3 (MaxPooling2D)	(None, 16, 16, 64)	0
conv2d_4 (Conv2D)	(None, 14, 14, 64)	36928
max_pooling2d_4 (MaxPooling2D)	(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

```
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')
```

4

Found 1027 images belonging to 2 classes.

2

5

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

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

6

```
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")
```

7

Using generated or real-world images

```
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")
```

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](#).

```
# 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
```

Using generated or real-world images

```
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 (MaxPooling2D)	(None, 73, 73, 32)
conv2d_2 (Conv2D)	(None, 71, 71, 64)
i2 (None, 35, 35, 64)	
(None, 33, 33, 64)	
i2 (None, 16, 16, 64)	
(None, 14, 14, 64)	
i2 (None, 7, 7, 64)	
(None, 3136)	
(None, 512)	
(None, 1)	

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

conv2d



max_pooling2d



conv2d_1



max_pooling2d_1



conv2d_2



max_pooling2d_2



conv2d_3



max_pooling2d_3



conv2d_4



```
model.layers
```

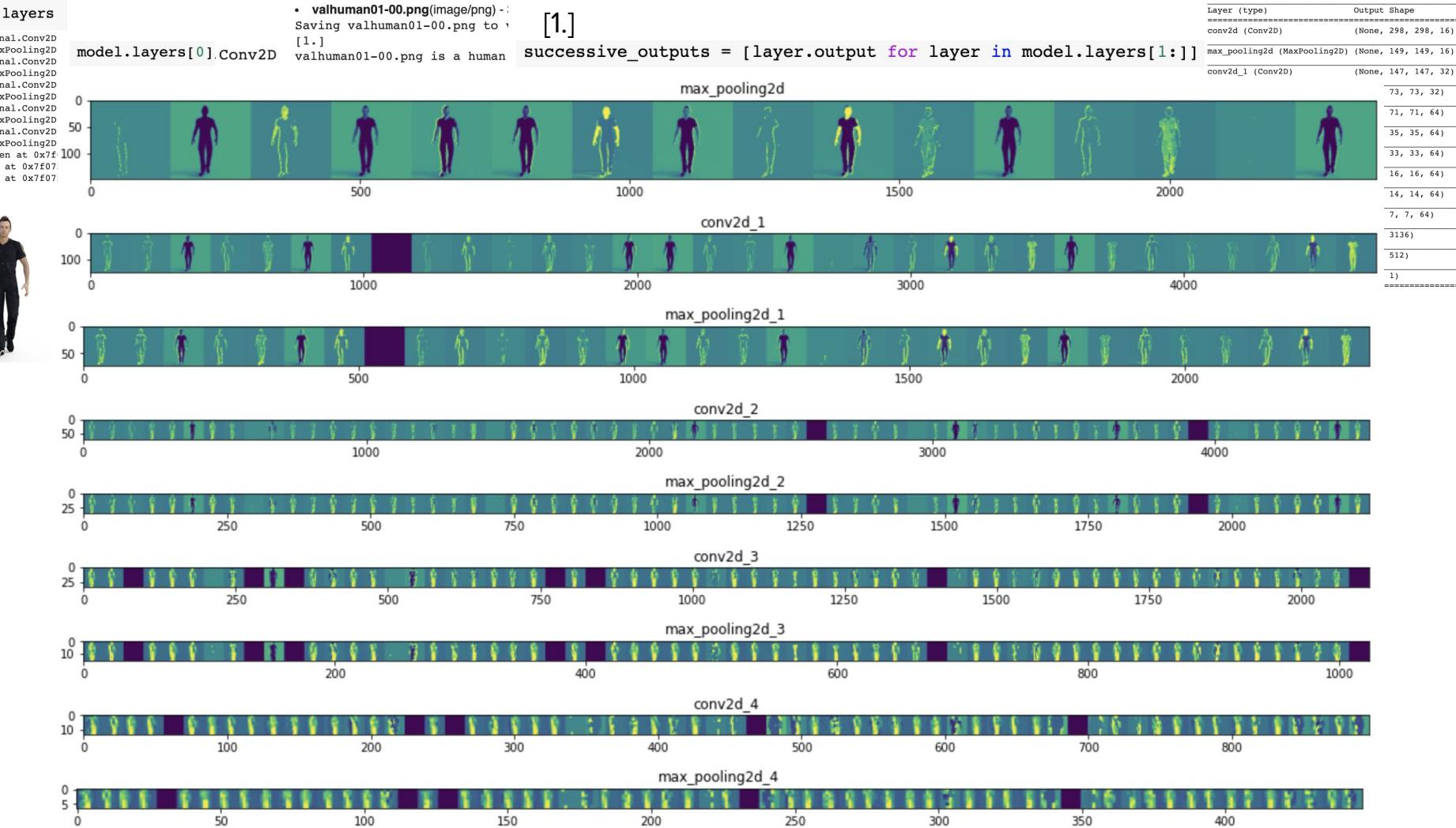
- valhuman01-00.png(image/png) -:
Saving valhuman01-00.png to
[1.]

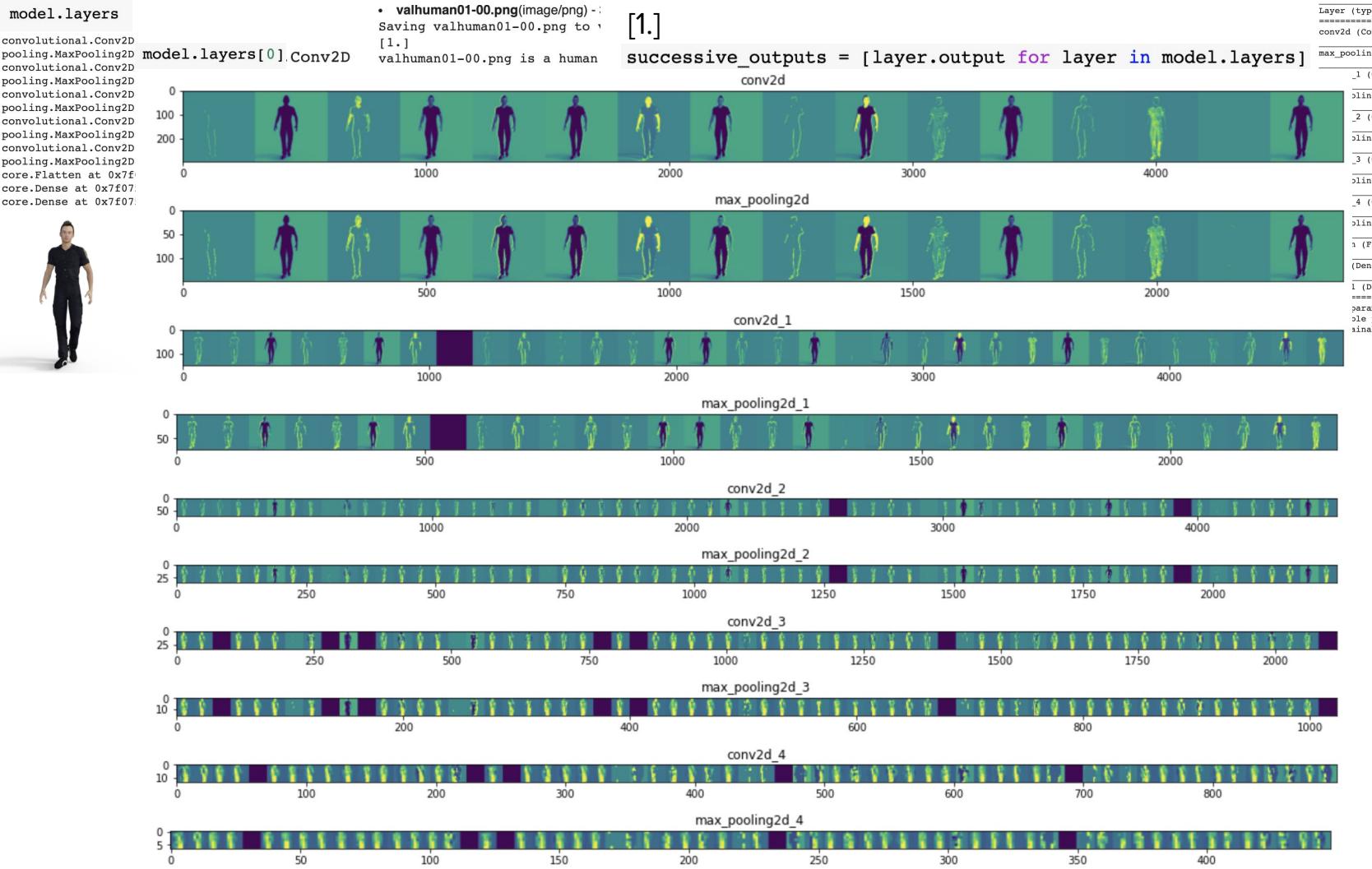
[1.]

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



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





```
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 0x7f07  
core.Dense at 0x7f07  
core.Dense at 0x7f07
```



```
• 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]

conv2d



max_pooling2d



conv2d_1



max_pooling2d_1



conv2d_2



max_pooling2d_2



conv2d_3



max_pooling2d_3



conv2d_4



max_pooling2d_4



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 (MaxPooling2D)	(None, 73, 73, 32)
l (Conv2D)	(None, 71, 71, 64)
ling2d_2 (MaxPooling2D)	(None, 35, 35, 64)
l (Conv2D)	(None, 33, 33, 64)
ling2d_3 (MaxPooling2D)	(None, 16, 16, 64)
l (Conv2D)	(None, 14, 14, 64)
ling2d_4 (MaxPooling2D)	(None, 7, 7, 64)
(Flatten)	(None, 3136)
Dense	(None, 512)
(Dense)	(None, 1)

inputs: 1,704,097
tot params: 1,704,097
trainable 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 0x1f07  
core.Dense at 0x1f07  
core.Dense at 0x1f07
```



model.layers[0].Conv2D

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

[0.03831019]

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

conv2d



max_pooling2d



conv2d_1



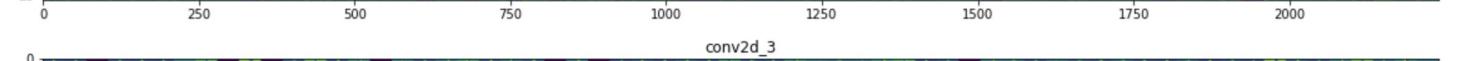
max_pooling2d_1



conv2d_2



max_pooling2d_2



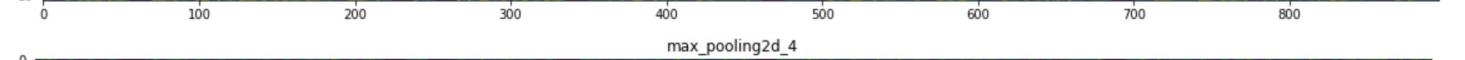
conv2d_3



max_pooling2d_3



conv2d_4



max_pooling2d_4



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 (MaxPooling2D)	(None, 73, 73, 32)
conv2d_2 (Conv2D)	(None, 71, 71, 64)
max_pooling2d_2 (MaxPooling2D)	(None, 35, 35, 64)
conv2d_3 (Conv2D)	(None, 33, 33, 64)
max_pooling2d_3 (MaxPooling2D)	(None, 16, 16, 64)
conv2d_4 (Conv2D)	(None, 14, 14, 64)
max_pooling2d_4 (MaxPooling2D)	(None, 7, 7, 64)
flatten (Flatten)	(None, 3136)
dense (Dense)	(None, 512)
dense (Dense)	(None, 1)

params: 1,704,097

base params: 1,704,097

trainable params: 0

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 (MaxPooling2D)	(None, 73, 73, 32)	0
conv2d_2 (Conv2D)	(None, 71, 71, 64)	18496
max_pooling2d_2 (MaxPooling2D)	(None, 35, 35, 64)	0
conv2d_3 (Conv2D)	(None, 33, 33, 64)	36928
max_pooling2d_3 (MaxPooling2D)	(None, 16, 16, 64)	0
conv2d_4 (Conv2D)	(None, 14, 14, 64)	36928
max_pooling2d_4 (MaxPooling2D)	(None, 7, 7, 64)	0
flatten (Flatten)	(None, 3136)	0
dense (Dense)	(None, 512)	1606144
dense_1 (Dense)	(None, 1)	513
<hr/>		
Total params:	1,704,097	2s 951ms/step - loss: 0.1559 - acc: 0.9445 - val_loss: 0.9205 - val_acc: 0.8711
Trainable params:	1,704,097	2s 210ms/step - loss: 1.3037 - acc: 0.8477
Non-trainable params:	0	9s 966ms/step - loss: 0.3236 - acc: 0.9114 - val_loss: 1.3037 - val_acc: 0.8477
<hr/>		
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

```

Using Real-world Images

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

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]

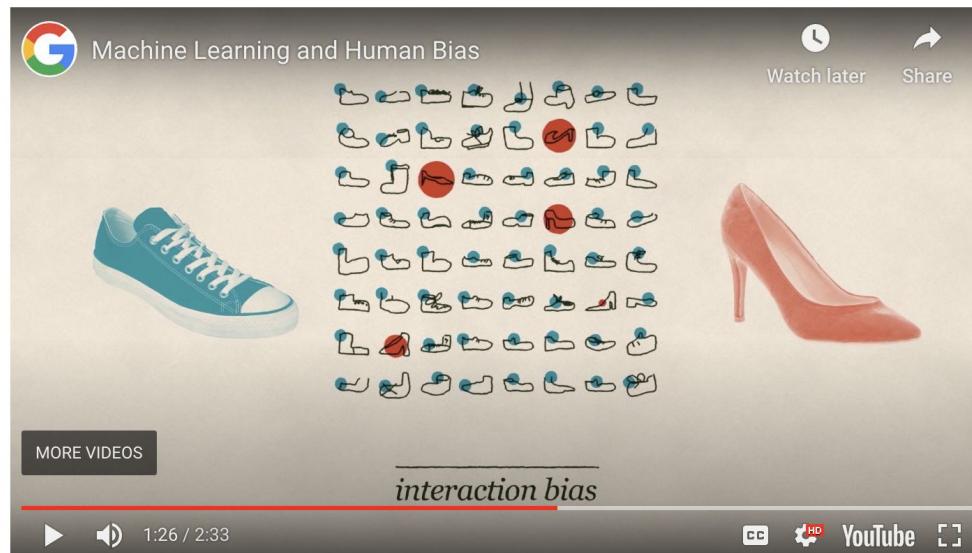
Remember the batch size you used when you created it, it was 20, that's important in the next step. There are 1,024 images in the training directory, so we're loading them in 128 at a time. So in order to load them all, we need to do 8 batches. So we set the steps_per_epoch to cover that.

```
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')
```

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/>

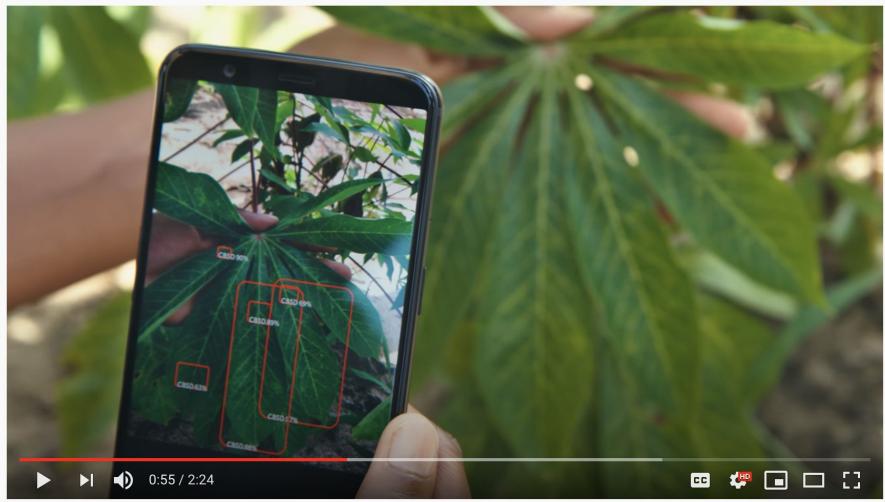
Check out these resources



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 ...

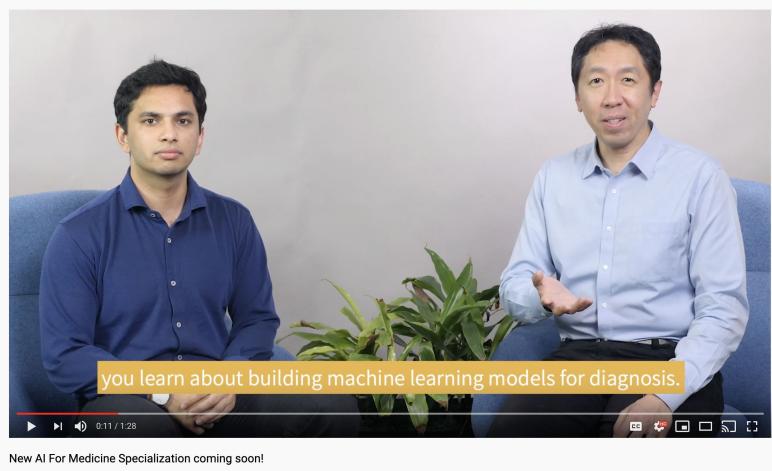
Source: <https://www.youtube.com/watch?v=NlpS-DhayQA>

<https://plantvillage.psu.edu/>

<https://www.iita.org/>

<https://www.blog.google/technology/ai/ai-takes-root-helping-farmers-identify-diseased-plants/>

Check out these events and Meetups



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

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

FRI, MAY 1, 7:30 PM EDT

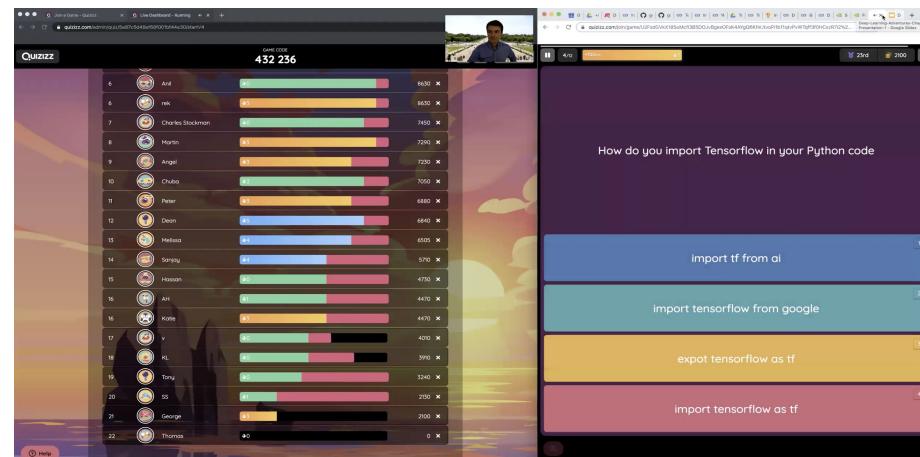
TensorFlow in Practice - Course 2 - Week 1,2 - Kaggle Challenge and Augmentation

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



The screenshot shows a Quizziz quiz interface. On the left, a scoreboard lists 22 participants with their names and scores. The top participant has a score of 432,236. On the right, there is a poll question: "How do you import Tensorflow in your Python code". Below the poll are four options: "import tf from ai", "import tensorflow from google", "export tensorflow as tf", and "import tensorflow as tf".

Rank	Name	Score
1	Andi	1630
2	ref	1630
3	Charles Stockman	1610
4	Martin	1540
5	Angel	1530
6	Chubs	1500
7	Peter	1480
8	Dean	1460
9	Helissa	1450
10	Serena	1410
11	Hossien	1370
12	Ari	1360
13	Kate	1340
14	v	1300
15	KL	1290
16	Toru	1280
17	SS	1250
18	George	1200
19	Thomas	1100
20		0

Time for a fun game



To play this game

1. Use any device to open
joinmyquiz.com
2. Enter game code
775667

or share via...

START

Your Quizizz name is... i

refresh icon

Start game

Game settings

Music **Sound effects**

Read aloud



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/5e87bdb07fa7f001b120404/start?from=soloLinkShare&referrer=5d921444d0fa99001a135336

The image shows two side-by-side screenshots. On the left is the Quizizz Live Dashboard for a game with the code 432 236. It displays a list of 22 players with their names, profile icons, scores, and progress bars. A video feed of a person is visible in the top right corner. On the right is a slide from a Google Slides presentation titled "Deep Learning Adventures - Chapter 1 - Presentation 1". The slide has a purple header with the text "How do you import Tensorflow in your Python code". Below the header are four numbered options: 1. "import tf from ai", 2. "import tensorflow from google", 3. "expot tensorflow as tf", and 4. "import tensorflow as tf".

Rank	Name	Score
6	Anil	8630
6	rek	8630
7	Charles Stockman	7450
8	Martin	7290
9	Angel	7230
10	Chuba	7050
11	Peter	6880
12	Dean	6840
13	Melissa	6505
14	Sanjay	5710
15	Hasson	4730
16	AH	4470
16	Kottie	4470
17	v	4010
18	KL	3910
19	Tony	3240
20	SS	2130
21	George	2100
22	Thomas	0

Questions

Discussion

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

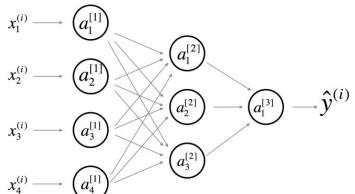


Figure 1: Comprehensive Network: representation commonly used for Neural Networks. For better aesthetic, we omitted the details on the parameters ($w_{ij}^{[l]}$ and $b_i^{[l]}$ etc...) that should appear on the edges

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:

$\cdot m$: number of examples in the dataset

$\cdot n_x$: input size

$\cdot n_y$: output size (or number of classes)

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

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

$\cdot L$: number of layers in the network.

Objects:

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

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

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

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

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

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

$\cdot \hat{y} \in \mathbb{R}^{n_y}$ 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

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

· 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:

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

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

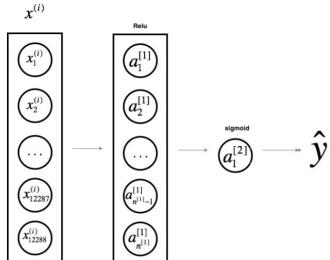


Figure 2: Simplified Network: a simpler representation of a two layer neural network, both are equivalent.

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. $\text{sign}(\delta)$). 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.

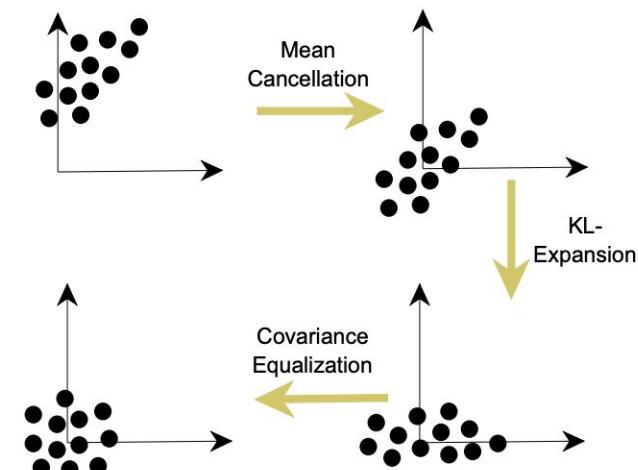
Transforming the Inputs

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.

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



Yann LeCun
@ylecun

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