PYIMAGESEARCH

DEEP LEARNING (HTTPS://WWW.PYIMAGESEARCH.COM/CATEGORY/DEEP-LEARNING/)

KERAS AND TENSORFLOW (HTTPS://WWW.PYIMAGESEARCH.COM/CATEGORY/KERAS-AND-TENSORFLOW/)

TUTORIALS (HTTPS://WWW.PYIMAGESEARCH.COM/CATEGORY/TUTORIALS/)

Video classification with Keras and Deep Learning

by **Adrian Rosebrock (https://www.pyimagesearch.com/author/adrian/)** *on* **July 15, 2019**

Click here to download the source code to this post (//app.monstercampaigns.com/c/tortsem7qkvyuxc4cyfi)

In this tutorial, you will learn how to perform **video classification using Keras, Python, and Deep Learning.**

Specifically, you will learn:

- The difference between video classification and standard image classification
- How to train a Convolutional Neural Network using Keras for image classification
- How to take that CNN and then **use it for video classification**
- How to **use rolling prediction averaging to reduce "flickering" in results**

This tutorial will serve as an introduction to the concept of working with deep learning in a temporal nature, paving the way for when we discuss Long Short-term Memory networks (LSTMs) and eventually human activity recognition.

To learn how to perform video classification with Keras and Deep learning, *just keep reading!*

Looking for the source code to this post?

JUMP RIGHT TO THE DOWNLOADS SECTION →

Video Classification with Keras and Deep Learning

Videos can be understood as a series of individual images; and therefore, many deep learning practitioners would be quick to treat video classification as performing image classification a total of *N* times, where *N* is the total number of frames in a video.

There's a problem with that approach though.

Video classification is *more* than just simple image classification — **with video we can typically make the assumption that subsequent frames in a video are *correlated* with respect to their *semantic contents*.**

If we are able to take advantage of the temporal nature of videos, we can improve our actual video classification results.

Neural network architectures such as Long short-term memory (LSTMs) and Recurrent Neural Networks (RNNs) are suited for time series data — two topics that we'll be covering in later tutorials — but in some cases, they may be overkill. They are also resource-hungry and time-consuming when it comes to training over thousands of *video* files as you can imagine.

Instead, for some applications, all you may need is *rolling averaging* over predictions.

In the remainder of this tutorial, you'll learn how to train a CNN for *image classification* (specifically sports classification) and then turn it into a more accurate *video classifier* by employing rolling averaging.

How is video classification different than image classification?

When performing image classification, we:

1. Input an image to our CNN
2. Obtain the predictions from the CNN
3. Choose the label with the largest corresponding probability

Since a video is just a series of frames, a naive video classification method would be to:

1. Loop over all frames in the video file
2. For each frame, pass the frame through the CNN
3. Classify each frame *individually* and *independently* of each other
4. Choose the label with the largest corresponding probability
5. Label the frame and write the output frame to disk

There's a problem with this approach though — if you've ever tried to apply simple image classification to video classification you likely encountered a sort of **"prediction flickering"** as seen in the video at the top of this section. Notice how in this visualization we see our CNN shifting between two predictions: *"football"* and the correct label, *"weight_lifting"*.

The video is clearly of weightlifting and we would like our entire video to be labeled as such — but how we can prevent the CNN "flickering" between these two labels?

A simple, yet elegant solution, is to utilize a rolling prediction average.

Our algorithm now becomes:

1. Loop over all frames in the video file
2. For each frame, pass the frame through the CNN
3. Obtain the predictions from the CNN
4. Maintain a list of the last *K* predictions
5. Compute the average of the last *K* predictions and choose the label with the largest corresponding probability
6. Label the frame and write the output frame to disk

The results of this algorithm can be seen in the video at the very top of this post — notice how the prediction flickering is gone and the entire video clip is correctly labeled!

In the remainder of this tutorial, you will learn how to implement this algorithm for video classification with Keras.

The Sports Classification Dataset

(https://pyimagesearch.com/wp-content/uploads/2019/07/keras_video_classification_sports_dataset.jpg)

Figure 1: A sports dataset curated by **GitHub user "anubhavmaity" (https://github.com/anubhavmaity)** using **Google Image Search (https://pyimagesearch.com/2017/12/04/how-to-create-a-deep-learning-dataset-using-google-images/)**. We will use this image dataset for video classification with Keras. (**image source (https://github.com/anubhavmaity/Sports-Type-Classifier/blob/master/readme_images/sports.png)**)

The dataset we'll be using here today is for sport/activity classification. The dataset was curated by **Anubhav Maity (https://github.com/anubhavmaity)** by downloading photos from **Google Images (https://pyimagesearch.com/2017/12/04/how-to-create-a-deep-learning-dataset-using-google-images/)** (**you could also use Bing (https://pyimagesearch.com/2018/04/09/how-to-quickly-build-a-deep-learning-image-dataset/)**) for the following categories:

1. Swimming
2. Badminton
3. Wrestling
4. Olympic Shooting
5. Cricket
6. Football
7. Tennis
8. Hockey
9. Ice Hockey
10. Kabaddi
11. WWE
12. Gymnasium
13. Weight lifting
14. Volleyball
15. Table tennis
16. Baseball
17. Formula 1
18. Moto GP
19. Chess
20. Boxing
21. Fencing
22. Basketball

To save time, computational resources, and to demonstrate the actual video classification algorithm (the actual point of this tutorial), we'll be training on a subset of the sports type dataset:

- **Football (i.e., soccer):** 799 images
- **Tennis:** 718 images
- **Weightlifting:** 577 images

Let's go ahead and download our dataset!

Downloading the Sports Classification Dataset

Go ahead and download the source code for today's blog post from the ***"Downloads"*** link.

Extract the .zip and navigate into the project folder from your terminal:

```
Video classification with Keras and Deep Learning
$ unzip keras-video-classification.zip
$ cd keras-video-classification
```

From there, clone **Anubhav Maity's (https://github.com/anubhavmaity)** repo:

```
Video classification with Keras and Deep Learning
$ git clone https://github.com/anubhavmaity/Sports-Type-Classifier
```

The data we'll be using today is now in the following path:

```
Video classification with Keras and Deep Learning
$ ls Sports-Type-Classifier/data | grep -Ev "urls|models|csv|pkl"
badminton
baseball
basketball
boxing
chess
cricket
fencing
football
formula1
gymnastics
hockey
ice_hockey
kabaddi
motogp
shooting
swimming
table_tennis
tennis
volleyball
weight_lifting
wrestling
wwe
```

Project Structure

Now that we have our project folder and **Anubhav Maity (https://github.com/anubhavmaity)**'s repo sitting inside, let's review our project structure:

```
Video classification with Keras and Deep Learning
$ tree --dirsfirst --filelimit 50
.
├── Sports-Type-Classifier
|   ├── data
|   |   ├── badminton [938 entries]
|   |   ├── baseball [746 entries]
|   |   ├── basketball [495 entries]
|   |   ├── boxing [705 entries]
|   |   ├── chess [481 entries]
|   |   ├── cricket [715 entries]
```

```
|   |   |   ├── fencing [635 entries]
|   |   |   ├── football [799 entries]
|   |   |   ├── formula1 [687 entries]
|   |   |   ├── gymnastics [719 entries]
|   |   |   ├── hockey [572 entries]
|   |   |   ├── ice_hockey [715 entries]
|   |   |   ├── kabaddi [454 entries]
|   |   |   ├── motogp [679 entries]
|   |   |   ├── shooting [536 entries]
|   |   |   ├── swimming [689 entries]
|   |   |   ├── table_tennis [713 entries]
|   |   |   ├── tennis [718 entries]
|   |   |   ├── volleyball [713 entries]
|   |   |   ├── weight_lifting [577 entries]
|   |   |   ├── wrestling [611 entries]
|   |   |   ├── wwe [671 entries]
|   |   ...
|   ├── example_clips
|   |   ├── lifting.mp4
|   |   ├── soccer.mp4
|   |   └── tennis.mp4
|   ├── model
|   |   ├── activity.model
|   |   └── lb.pickle
|   ├── output
|   ├── plot.png
|   ├── predict_video.py
|   └── train.py
|
|   29 directories, 41 files
```

Our training image data is in the `Sports-Type-Classifier/data/` directory, organized by class. There is additional clutter included with the GitHub repo that we won't be using. I've omitted it from the project structure output above since we only care about the data. *Furthermore, our training script will only train with football, tennis, and weightlifting data* (however a simple list item change could allow you to train with other classes as well).

I've extracted three `example_clips/` for us from YouTube to test our model upon. Credits for the three clips are at the bottom of the *"Keras video classification results"* section.

Our classifier files are in the `model/` directory. Included are `activity.model` (the trained Keras model) and `lb.pickle` (our label binarizer).

An empty `output/` folder is the location where we'll store video classification results.

We'll be covering two Python scripts in today's tutorial:

- `train.py` : A Keras training script that grabs the dataset class images that we care about, loads the **ResNet50** CNN, and applies **transfer learning/fine-tuning (https://pyimagesearch.com/2019/06/03/fine-tuning-with-keras-and-deep-learning/)** of ImageNet weights to train our model. The training script generates/outputs three files:
  - `model/activity.model` : A fine-tuned classifier based on ResNet50 for recognizing sports.
  - `model/lb.pickle` : A serialized label binarizer containing our unique class labels.
  - `plot.png` : The accuracy/loss training history plot.
- `predict_video.py` : Loads an input video from the `example_clips/` and proceeds to classify the video ideally using today's rolling average method.

Implementing our Keras training script

Let's go ahead and implement our training script used to train a Keras CNN to recognize each of the sports activities.

Open up the `train.py` file and insert the following code:

Video classification with Keras and Deep Learning

```
1.  # set the matplotlib backend so figures can be saved in the background
2.  import matplotlib
3.  matplotlib.use("Agg")
4.
5.  # import the necessary packages
6.  from keras.preprocessing.image import ImageDataGenerator
7.  from keras.layers.pooling import AveragePooling2D
8.  from keras.applications import ResNet50
9.  from keras.layers.core import Dropout
10. from keras.layers.core import Flatten
11. from keras.layers.core import Dense
12. from keras.layers import Input
13. from keras.models import Model
14. from keras.optimizers import SGD
15. from sklearn.preprocessing import LabelBinarizer
16. from sklearn.model_selection import train_test_split
17. from sklearn.metrics import classification_report
18. from imutils import paths
19. import matplotlib.pyplot as plt
20. import numpy as np
21. import argparse
22. import pickle
23. import cv2
24. import os
```

On **Lines 2-24**, we import necessary packages for training our classifier:

- `matplotlib` : For plotting. **Line 3** sets the backend so we can output our training plot to a .png image file.
- `keras` : For deep learning. Namely, we'll use the `ResNet50` CNN. We'll also work with the `ImageDataGenerator` which you can read about in **last week's tutorial (https://pyimagesearch.com/2019/07/08/keras-**

imagedatagenerator-and-data-augmentation/).

- `sklearn` : From scikit-learn we'll use their implementation of a `LabelBinarizer` for one-hot encoding our class labels. The `train_test_split` function will segment our dataset into training and testing splits. We'll also print a `classification_report` in a traditional format.
- `paths` : Contains convenience functions for listing all image files in a given path. From there we'll be able to load our images into memory.
- `numpy` : Python's *de facto* numerical processing library.
- `argparse` : For parsing **command line arguments (https://pyimagesearch.com/2018/03/12/python-argparse-command-line-arguments/)**.
- `pickle` : For serializing our label binarizer to disk.
- `cv2` : OpenCV.
- `os` : The operating system module will be used to ensure we grab the correct file/path separator which is OS-dependent.

Let's go ahead and **parse our command line arguments (https://pyimagesearch.com/2018/03/12/python-argparse-command-line-arguments/)** now:

```
Video classification with Keras and Deep Learning
# construct the argument parser and parse the arguments
ap = argparse.ArgumentParser()
ap.add_argument("-d", "--dataset", required=True,
```

```
    help="path to input dataset")
ap.add_argument("-m", "--model", required=True,
    help="path to output serialized model")
ap.add_argument("-l", "--label-bin", required=True,
    help="path to output label binarizer")
ap.add_argument("-e", "--epochs", type=int, default=25,
    help="# of epochs to train our network for")
ap.add_argument("-p", "--plot", type=str, default="plot.png",
    help="path to output loss/accuracy plot")
args = vars(ap.parse_args())
```

Our script accepts five command line arguments, the first three of which are required:

- `--dataset` : The path to the input dataset.
- `--model` : Our path to our output Keras model file.
- `--label-bin` : The path to our output label binarizer pickle file.
- `--epochs` : How many epochs to train our network for — by default, we'll train for `25` epochs, but as I'll show later in the tutorial, `50` epochs can lead to better results.
- `--plot` : The path to our output plot image file — by default it will be named `plot.png` and be placed in the same directory as this training script.

With our command line arguments parsed and in-hand, let's proceed to initialize our `LABELS` and load our `data` :

```
Video classification with Keras and Deep Learning
# initialize the set of labels from the spots activity dataset we are
# going to train our network on
LABELS = set(["weight_lifting", "tennis", "football"])

# grab the list of images in our dataset directory, then initialize
# the list of data (i.e., images) and class images
print("[INFO] loading images...")
```

```
47. imagePaths = list(paths.list_images(args["dataset"]))
48. data = []
49. labels = []
50.
51. # loop over the image paths
52. for imagePath in imagePaths:
53.    # extract the class label from the filename
54.    label = imagePath.split(os.path.sep)[-2]
55.
56.    # if the label of the current image is not part of of the labels
57.    # are interested in, then ignore the image
58.    if label not in LABELS:
59.        continue
60.
61.    # load the image, convert it to RGB channel ordering, and resize
62.    # it to be a fixed 224x224 pixels, ignoring aspect ratio
63.    image = cv2.imread(imagePath)
64.    image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
65.    image = cv2.resize(image, (224, 224))
66.
67.    # update the data and labels lists, respectively
68.    data.append(image)
69.    labels.append(label)
```

Line 42 contains the set of class `LABELS` for which our dataset will consist of. All labels *not* present in this set will be *excluded* from being part of our dataset. To save on training time, our dataset will only consist of *weight lifting, tennis,* and *football/soccer*. Feel free to work with other classes by making changes to the `LABELS` set.

All dataset `imagePaths` are gathered via **Line 47** and the value contained in `args["dataset"]` (which comes from our **command line arguments (https://pyimagesearch.com/2018/03/12/python-argparse-command-line-arguments/)**).

Lines 48 and 49 initialize our `data` and `labels` lists.

From there, we'll begin looping over all `imagePaths` on **Line 52**.

In the loop, first we extract the class `label` from the `imagePath` (**Line 54**). **Lines 58 and 59** then ignore any `label` not in the `LABELS` set.

Lines 63-65 load and preprocess an `image`. Preprocessing includes swapping color channels for OpenCV to Keras compatibility and resizing to *224×224*px. Read more about **resizing images for CNNs here (https://pyimagesearch.com/2019/06/24/change-input-shape-dimensions-for-fine-tuning-with-keras/)**. To learn more about the importance of preprocessing be sure to refer to ***Deep Learning for Computer Vision with Python* (https://pyimagesearch.com/deep-learning-computer-vision-python-book/)**.

The `image` and `label` are then added to the `data` and `labels` lists, respectively on **Lines 68 and 69**.

Continuing on, we will one-hot encode our `labels` and partition our `data`:

Video classification with Keras and Deep Learning

```
71. # convert the data and labels to NumPy arrays
72. data = np.array(data)
73. labels = np.array(labels)
74.
75. # perform one-hot encoding on the labels
76. lb = LabelBinarizer()
77. labels = lb.fit_transform(labels)
78.
79. # partition the data into training and testing splits using 75% of
80. # the data for training and the remaining 25% for testing
81. (trainX, testX, trainY, testY) = train_test_split(data, labels,
82.     test_size=0.25, stratify=labels, random_state=42)
```

Lines 72 and 73 convert our `data` and `labels` lists into NumPy arrays.

One-hot encoding of `labels` takes place on **Lines 76 and 77**. One-hot encoding is a way of marking an active class label via binary array elements. For example "football" may be `array([1, 0, 0])` whereas "weightlifting" may be `array([0, 0, 1])`. Notice how only one class is "hot" at any given time.

Lines 81 and 82 then segment our `data` into training and testing splits using 75% of the data for training and the remaining 25% for testing.

Let's initialize our **data augmentation (https://pyimagesearch.com/2019/07/08/keras-imagedatagenerator-and-data-augmentation/)** object:

Video classification with Keras and Deep Learning

```
84. # initialize the training data augmentation object
85. trainAug = ImageDataGenerator(
86.    rotation_range=30,
87.    zoom_range=0.15,
88.    width_shift_range=0.2,
89.    height_shift_range=0.2,
90.    shear_range=0.15,
91.    horizontal_flip=True,
92.    fill_mode="nearest")
93.
94. # initialize the validation/testing data augmentation object (which
95. # we'll be adding mean subtraction to)
96. valAug = ImageDataGenerator()
97.
98. # define the ImageNet mean subtraction (in RGB order) and set the
99. # the mean subtraction value for each of the data augmentation
100. # objects
101. mean = np.array([123.68, 116.779, 103.939], dtype="float32")
102. trainAug.mean = mean
103. valAug.mean = mean
```

Lines 85-96 initialize two data augmentation objects — one for training and one for validation. Data augmentation is nearly always recommended in deep learning for computer vision to increase model generalization.

The `trainAug` object performs random rotations, zooms, shifts, shears, and flips on our data. You can **read more about the `ImageDataGenerator` and `fit_generator` here (https://pyimagesearch.com/2018/12/24/how-to-use-keras-fit-and-fit_generator-a-hands-on-tutorial/)**. As we reinforced last week, keep in mind that **with Keras, images will be generated on-the-fly (https://pyimagesearch.com/2019/07/08/keras-imagedatagenerator-and-data-augmentation/)** (it is not an additive operation).

No augmentation will be conducted for validation data (`valAug`), but we will perform mean subtraction.

The `mean` pixel value is set on **Line 101**. From there, **Lines 102 and 103** set the `mean` attribute for `trainAug` and `valAug` so that mean subtraction will be conducted as images are generated during training/evaluation.

Now we're going to perform what I like to call "network surgery" as part of **fine-tuning (https://pyimagesearch.com/2019/06/03/fine-tuning-with-keras-and-deep-learning/)**:

```
Video classification with Keras and Deep Learning
# load the ResNet-50 network, ensuring the head FC layer sets are left
# off
baseModel = ResNet50(weights="imagenet", include_top=False,
    input_tensor=Input(shape=(224, 224, 3)))
```

```
109. |
110. |  # construct the head of the model that will be placed on top of the
111. |  # the base model
112. |  headModel = baseModel.output
113. |  headModel = AveragePooling2D(pool_size=(7, 7))(headModel)
114. |  headModel = Flatten(name="flatten")(headModel)
115. |  headModel = Dense(512, activation="relu")(headModel)
116. |  headModel = Dropout(0.5)(headModel)
117. |  headModel = Dense(len(lb.classes_), activation="softmax")(headModel)
118. |
119. |  # place the head FC model on top of the base model (this will become
120. |  # the actual model we will train)
121. |  model = Model(inputs=baseModel.input, outputs=headModel)
122. |
123. |  # loop over all layers in the base model and freeze them so they will
124. |  # *not* be updated during the training process
125. |  for layer in baseModel.layers:
126. |     layer.trainable = False
```

Lines 107 and 108 load `ResNet50` pre-trained with ImageNet weights while chopping the head of the network off.

From there, **Lines 112-121** assemble a new `headModel` and suture it onto the `baseModel` .

We'll now freeze the `baseModel` so that it will ***not*** be trained via backpropagation (**Lines 125 and 126**).

Let's go ahead and compile + train our `model` :

```
Video classification with Keras and Deep Learning
128. |  # compile our model (this needs to be done after our setting our
129. |  # layers to being non-trainable)
130. |  print("[INFO] compiling model...")
131. |  opt = SGD(lr=1e-4, momentum=0.9, decay=1e-4 / args["epochs"])
132. |  model.compile(loss="categorical_crossentropy", optimizer=opt,
133. |     metrics=["accuracy"])
134. |
135. |  # train the head of the network for a few epochs (all other layers
136. |  # are frozen) -- this will allow the new FC layers to start to become
137. |  # initialized with actual "learned" values versus pure random
138. |  print("[INFO] training head...")
139. |  H = model.fit_generator(
140. |     trainAug.flow(trainX, trainY, batch_size=32),
```

```
141.     steps_per_epoch=len(trainX) // 32,
142.     validation_data=valAug.flow(testX, testY),
143.     validation_steps=len(testX) // 32,
144.     epochs=args["epochs"])
```

Lines 131-133 `compile` our `model` with the Stochastic Gradient Descent (`SGD`) optimizer with an initial learning rate of `1e-4` and learning rate decay. We use `"categorical_crossentropy"` loss for training with multiple classes. If you are working with only two classes, be sure to use `"binary_crossentropy"` loss.

A call to the `fit_generator` function on our `model` (**Lines 139-144**) trains our network with data augmentation and mean subtraction.

Keep in mind that our `baseModel` is frozen and we're only training the head. This is known as "fine-tuning". For a quick overview of fine-tuning, be sure to read **my previous article (https://pyimagesearch.com/2019/06/03/fine-tuning-with-keras-and-deep-learning/)**. And for a more in-depth dive into fine-tuning, pick up a copy of the *Practitioner Bundle* of ***Deep Learning for Computer Vision with Python* (https://pyimagesearch.com/deep-learning-computer-vision-python-book/)**.

We'll begin to wrap up by evaluating our network and plotting the training history:

```
Video classification with Keras and Deep Learning
146.  # evaluate the network
147.  print("[INFO] evaluating network...")
148.  predictions = model.predict(testX, batch_size=32)
149.  print(classification_report(testY.argmax(axis=1),
150.     predictions.argmax(axis=1), target_names=lb.classes_))
151.
152.  # plot the training loss and accuracy
153.  N = args["epochs"]
```

```
154. | plt.style.use("ggplot")
155. | plt.figure()
156. | plt.plot(np.arange(0, N), H.history["loss"], label="train_loss")
157. | plt.plot(np.arange(0, N), H.history["val_loss"], label="val_loss")
158. | plt.plot(np.arange(0, N), H.history["acc"], label="train_acc")
159. | plt.plot(np.arange(0, N), H.history["val_acc"], label="val_acc")
160. | plt.title("Training Loss and Accuracy on Dataset")
161. | plt.xlabel("Epoch #")
162. | plt.ylabel("Loss/Accuracy")
163. | plt.legend(loc="lower left")
164. | plt.savefig(args["plot"])
```

After we evaluate our network on the testing set and print a `classification_report` (**Lines 148-150**), we go ahead and plot our accuracy/loss curves with matplotlib (**Lines 153-163**). The plot is saved to disk via **Line 164**.

To wrap up will serialize our `model` and label binarizer (`lb`) to disk:

```
    Video classification with Keras and Deep Learning
166. | # serialize the model to disk
167. | print("[INFO] serializing network...")
168. | model.save(args["model"])
169. |
170. | # serialize the label binarizer to disk
171. | f = open(args["label_bin"], "wb")
172. | f.write(pickle.dumps(lb))
173. | f.close()
```

Line 168 saves our fine-tuned Keras `model` .

Finally, **Lines 171** serialize and store our label binarizer in Python's pickle format.

Training results

Before we can (1) classify frames in a video with our CNN and then (2) utilize our CNN for video classification, we first need to train the model.

Make sure you have used the ***"Downloads"*** section of this tutorial to download the source code to this image (as well as downloaded the sports type dataset).

From there, open up a terminal and execute the following command:

```
Video classification with Keras and Deep Learning
 $ python train.py --dataset Sports-Type-Classifier/data --model
output/activity.model \
    --label-bin output/lb.pickle --epochs 50
 [INFO] loading images...
 [INFO] compiling model...
 [INFO] training head...
 Epoch 1/50
 48/48 [==============================] - 21s 445ms/step - loss: 1.1552 - acc:
0.4329 - val_loss: 0.7308 - val_acc: 0.6699
 Epoch 2/50
 48/48 [==============================] - 18s 368ms/step - loss: 0.9412 - acc:
0.5801 - val_loss: 0.5987 - val_acc: 0.7346
 Epoch 3/50
 48/48 [==============================] - 17s 351ms/step - loss: 0.8054 - acc:
0.6504 - val_loss: 0.5181 - val_acc: 0.7613
 Epoch 4/50
 48/48 [==============================] - 17s 353ms/step - loss: 0.7215 - acc:
0.6966 - val_loss: 0.4497 - val_acc: 0.7922
 Epoch 5/50
 48/48 [==============================] - 17s 353ms/step - loss: 0.6253 - acc:
0.7572 - val_loss: 0.4530 - val_acc: 0.7984
 ...
 Epoch 46/50
 48/48 [==============================] - 17s 352ms/step - loss: 0.2325 - acc:
0.9167 - val_loss: 0.2024 - val_acc: 0.9198
 Epoch 47/50
 48/48 [==============================] - 17s 349ms/step - loss: 0.2284 - acc:
0.9212 - val_loss: 0.2058 - val_acc: 0.9280
 Epoch 48/50
 48/48 [==============================] - 17s 348ms/step - loss: 0.2261 - acc:
0.9212 - val_loss: 0.2448 - val_acc: 0.9095
 Epoch 49/50
 48/48 [==============================] - 17s 348ms/step - loss: 0.2170 - acc:
0.9153 - val_loss: 0.2259 - val_acc: 0.9280
 Epoch 50/50
 48/48 [==============================] - 17s 352ms/step - loss: 0.2109 - acc:
0.9225 - val_loss: 0.2267 - val_acc: 0.9218
 [INFO] evaluating network...
                 precision    recall  f1-score   support

       football       0.86      0.98      0.92       196
         tennis       0.95      0.88      0.91       179
 weight_lifting       0.98      0.87      0.92       143
```

```
33. |
34. |       micro avg       0.92      0.92      0.92       518
35. |       macro avg       0.93      0.91      0.92       518
36. |    weighted avg       0.92      0.92      0.92       518
37. |
38. |  [INFO] serializing network...
```

(https://pyimagesearch.com/wp-content/uploads/2019/07/plot.png)

Figure 2: Sports video classification with Keras accuracy/loss training history plot.

As you can see, we're obtaining **~92-93% accuracy** after fine-tuning ResNet50 on the sports dataset.

Checking our model directory we can see that the fine-tuned model along with the label binarizer have been serialized to disk:

```
Video classification with Keras and Deep Learning
$ ls model/
activity.model lb.pickle
```

We'll then take these files and use them to implement rolling prediction averaging in the next section.

Video classification with Keras and rolling prediction averaging

We are now ready to implement video classification with Keras via rolling prediction accuracy!

To create this script we'll take advantage of the *temporal nature of videos*, specifically the assumption that *subsequent frames in a video will have similar semantic contents*.

By performing rolling prediction accuracy we'll be able to "smoothen out" the predictions and avoid "prediction flickering".

Let's get started — open up the `predict_video.py` file and insert the following code:

```
Video classification with Keras and Deep Learning
# import the necessary packages
from keras.models import load_model
from collections import deque
import numpy as np
import argparse
import pickle
import cv2

# construct the argument parser and parse the arguments
ap = argparse.ArgumentParser()
ap.add_argument("-m", "--model", required=True,
```

```
12.     help="path to trained serialized model")
13.  ap.add_argument("-l", "--label-bin", required=True,
14.     help="path to  label binarizer")
15.  ap.add_argument("-i", "--input", required=True,
16.     help="path to our input video")
17.  ap.add_argument("-o", "--output", required=True,
18.     help="path to our output video")
19.  ap.add_argument("-s", "--size", type=int, default=128,
20.     help="size of queue for averaging")
21.  args = vars(ap.parse_args())
```

Lines 2-7 load necessary packages and modules. In particular, we'll be using `deque` from Python's `collections` module to assist with our rolling average algorithm.

Then, **Lines 10-21** parse five **command line arguments (https://pyimagesearch.com/2018/03/12/python-argparse-command-line-arguments/)**, four of which are required:

- `--model` : The path to the input model generated from our previous training step.
- `--label-bin` : The path to the serialized pickle-format label binarizer generated by the previous script.
- `--input` : A path to an input video for video classification.
- `--output` : The path to our output video which will be saved to disk.
- `--size` : The max size of the queue for rolling averaging (`128` by default). For some of our example results later on, we'll set the size to `1` so that no averaging is performed.

Armed with our imports and command line `args`, we're now ready to perform initializations:

```
Video classification with Keras and Deep Learning
23. # load the trained model and label binarizer from disk
24. print("[INFO] loading model and label binarizer...")
25. model = load_model(args["model"])
26. lb = pickle.loads(open(args["label_bin"], "rb").read())
27.
28. # initialize the image mean for mean subtraction along with the
29. # predictions queue
30. mean = np.array([123.68, 116.779, 103.939][::1], dtype="float32")
31. Q = deque(maxlen=args["size"])
```

Lines 25 and 26 load our `model` and label binarizer.

Line 30 then sets our `mean` subtraction value.

We'll use a `deque` to implement our rolling prediction averaging. Our deque, `Q`, is initialized with a `maxlen` equal to the `args["size"]` value (**Line 31**).

Let's initialize our `cv2.VideoCapture` object and begin looping over video frames:

```
Video classification with Keras and Deep Learning
33. # initialize the video stream, pointer to output video file, and
34. # frame dimensions
35. vs = cv2.VideoCapture(args["input"])
36. writer = None
37. (W, H) = (None, None)
38.
39. # loop over frames from the video file stream
40. while True:
41.    # read the next frame from the file
42.    (grabbed, frame) = vs.read()
43.
44.    # if the frame was not grabbed, then we have reached the end
45.    # of the stream
46.    if not grabbed:
47.        break
48.
```

```
49.     # if the frame dimensions are empty, grab them
50.     if W is None or H is None:
51.         (H, W) = frame.shape[:2]
```

Line 35 grabs a pointer to our input video file stream. We use the `VideoCapture` class from OpenCV to read frames from our video stream.

Our video `writer` and dimensions are then initialized to `None` via **Lines 36 and 37**.

Line 40 begins our video classification `while` loop.

First, we grab a `frame` (**Lines 42-47**). If the `frame` was `not grabbed`, then we've reached the end of the video, at which point we'll `break` from the loop.

Lines 50-51 then set our frame dimensions if required.

Let's preprocess our `frame`:

```
    Video classification with Keras and Deep Learning
53.     # clone the output frame, then convert it from BGR to RGB
54.     # ordering, resize the frame to a fixed 224x224, and then
55.     # perform mean subtraction
56.     output = frame.copy()
57.     frame = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
58.     frame = cv2.resize(frame, (224, 224)).astype("float32")
59.     frame -= mean
```

A `copy` of our frame is made for `output` purposes (**Line 56**).

We then preprocess the `frame` using the same steps as our training script, including:

- Swapping color channels (**Line 57**).
- Resizing to *224×224*px (**Line 58**).
- Mean subtraction (**Line 59**).

Frame classification inference and *rolling prediction averaging* come next:

```
Video classification with Keras and Deep Learning
61.     # make predictions on the frame and then update the predictions
62.     # queue
63.     preds = model.predict(np.expand_dims(frame, axis=0))[0]
64.     Q.append(preds)
65.
66.     # perform prediction averaging over the current history of
67.     # previous predictions
68.     results = np.array(Q).mean(axis=0)
69.     i = np.argmax(results)
70.     label = lb.classes_[i]
```

Line 63 makes predictions on the *current* frame. The **prediction results are added to the** `Q` via **Line 64**.

From there, **Lines 68-70** perform **prediction averaging over the** `Q` history resulting in a class `label` for the rolling average. Broken down, these lines find the label with the largest corresponding probability across the average predictions.

Now that we have our resulting `label`, let's annotate our `output` frame and write it to disk:

```
Video classification with Keras and Deep Learning
72.     # draw the activity on the output frame
73.     text = "activity: {}".format(label)
74.     cv2.putText(output, text, (35, 50), cv2.FONT_HERSHEY_SIMPLEX,
75.         1.25, (0, 255, 0), 5)
```

```
76. |
77. |     # check if the video writer is None
78. |     if writer is None:
79. |         # initialize our video writer
80. |         fourcc = cv2.VideoWriter_fourcc(*"MJPG")
81. |         writer = cv2.VideoWriter(args["output"], fourcc, 30,
82. |             (W, H), True)
83. |
84. |     # write the output frame to disk
85. |     writer.write(output)
86. |
87. |     # show the output image
88. |     cv2.imshow("Output", output)
89. |     key = cv2.waitKey(1) & 0xFF
90. |
91. |     # if the `q` key was pressed, break from the loop
92. |     if key == ord("q"):
93. |         break
94. |
95. | # release the file pointers
96. | print("[INFO] cleaning up...")
97. | writer.release()
98. | vs.release()
```

Lines 73-75 draw the prediction on the `output` frame.

Lines 78-82 initialize the video `writer` if necessary. The `output` frame is written to the file (**Line 85**). Read more about **writing to video files with OpenCV here (https://pyimagesearch.com/2016/02/22/writing-to-video-with-opencv/)**.

The `output` is also displayed on the screen until the “q” key is pressed (or until the end of the video file is reached as aforementioned) via **Lines 88-93**.

Finally, we’ll perform cleanup (**Lines 97 and 98**).

Keras video classification results

Now that we've implemented our video classifier with Keras, let's put it to work.

Make sure you've used the ***"Downloads"*** section of this tutorial to download the source code.

From there, let's apply video classification to a "tennis" clip — but let's set the `--size` of the queue to `1`, trivially turning video classification into standard image classification:

```
Video classification with Keras and Deep Learning
$ python predict_video.py --model model/activity.model \
   --label-bin model/lb.pickle \
   --input example_clips/tennis.mp4 \
   --output output/tennis_1frame.avi \
   --size 1
Using TensorFlow backend.
[INFO] loading model and label binarizer...
[INFO] cleaning up...
```

0:08

As you can see, there is quite a bit of label flickering — our CNN thinks certain frames are “tennis” (correct) while others are “football” (incorrect).

Let’s now use the default queue `--size` of `128`, thus utilizing our **prediction averaging algorithm** to smoothen the results:

```
Video classification with Keras and Deep Learning
$ python predict_video.py --model model/activity.model \
   --label-bin model/lb.pickle \
   --input example_clips/tennis.mp4 \
   --output output/tennis_128frames_smoothened.avi \
   --size 128
Using TensorFlow backend.
[INFO] loading model and label binarizer...
[INFO] cleaning up...
```

0:08

Notice how we’ve correctly labeled this video as “tennis”!

Let's try a different example, this one of "weightlifting". Again, we'll start off by using a queue `--size` of `1`:

```
Video classification with Keras and Deep Learning
$ python predict_video.py --model model/activity.model \
   --label-bin model/lb.pickle \
   --input example_clips/lifting.mp4 \
   --output output/lifting_1frame.avi \
   --size 1
Using TensorFlow backend.
[INFO] loading model and label binarizer...
[INFO] cleaning up...
```

0:09

We once again encounter prediction flickering.

However, if we use a frame `--size` of `128`, our prediction averaging will obtain the desired result:

```
Video classification with Keras and Deep Learning
$ python predict_video.py --model model/activity.model \
```

```
    --label-bin model/lb.pickle \
    --input example_clips/lifting.mp4 \
    --output output/lifting_128frames_smoothened.avi \
    --size 128
Using TensorFlow backend.
[INFO] loading model and label binarizer...
[INFO] cleaning up...
```

Let's try one final example:

```
Video classification with Keras and Deep Learning
$ python predict_video.py --model model/activity.model \
    --label-bin model/lb.pickle \
    --input example_clips/soccer.mp4 \
    --output output/soccer_128frames_smoothened.avi \
    --size 128
Using TensorFlow backend.
[INFO] loading model and label binarizer...
[INFO] cleaning up...
```

Here you can see the input video is correctly classified as “football” (i.e., soccer).

Notice that there is no frame flickering — our rolling prediction averaging smoothes out the predictions.

While simple, this algorithm can enable you to perform video classification with Keras!

In future tutorials, we’ll cover more advanced methods of activity and video classification, including LSTMs and RNNs.

Video Credits:

- ***Ultimate Olympic Weightlifting Motivation (https://www.youtube.com/watch?v=vPBBI28MkCU)*** – Alex Yao
- ***The Best Game Of Tennis Ever? | Australian Open 2012***

(https://www.youtube.com/watch?v=oyxhHkOel2I) – Australian Open TV

- *Germany v Sweden – 2018 FIFA World Cup Russia™ – Match 27 (https://www.youtube.com/watch?v=4e9a3KptfC0)* – FIFATV

Summary

In this tutorial, you learned how to perform video classification with Keras and deep learning.

A naïve algorithm to video classification would be to treat each individual frame of a video as independent from the others. This type of implementation will cause "label flickering" where the CNN returns different labels for subsequent frames, *even though the frames should be the same labels!*

More advanced neural networks, including LSTMs and the more general RNNs, can help combat this problem and lead to much higher accuracy. However, LSTMs and RNNs can be dramatic overkill dependent on what you are doing — **in some situations, simple rolling prediction averaging will give you the results you need.**

Using rolling prediction averaging, you maintain a list of the last *K* predictions from the CNN. We then take these last *K* predictions, average them, select the label with the largest probability, and choose this label to classify the *current* frame. The assumption here is that subsequent frames in a video will have similar semantic contents.

If that assumption holds then we can take advantage of the temporal nature of videos, assuming that the previous frames are similar to the current frame.

The averaging, therefore, enables us to smooth out the predictions and make for a better video classifier.

In a future tutorial, we'll discuss the more advanced LSTMs and RNNs as well. But in the meantime, **take a look at this guide to deep learning action recognition. (http://blog.qure.ai/notes/deep-learning-for-videos-action-recognition-review)**

To download the source code to this post, and to be notified when future tutorials are published here on PyImageSearch, *just enter your email address in the form below!*

Download the Source Code and FREE 17-page Resource Guide

Enter your email address below to get a .zip of the code and a **FREE 17-page Resource Guide on Computer Vision, OpenCV, and Deep Learning.** Inside you'll find my hand-picked tutorials, books, courses, and libraries to help you master CV and DL!

About the Author

Hi there, I'm Adrian Rosebrock, PhD. All too often I see developers, students, and researchers wasting their time, studying the wrong things, and generally struggling to get started with Computer Vision, Deep Learning, and OpenCV. I created this website to show you what I believe is the best possible way to get your start.

Previous Article:
Keras ImageDataGenerator and Data Augmentation
(https://www.pyimagesearch.com/2019/07/08/keras-imagedatagenerator-and-data-augmentation/)

Next Article:
Keras learning ra
(https://www.pyimage
learning-rate

170 responses to: Video classification with Keras and Deep Learning

Neeraj Sujan
July 15, 2019 at 11:37 am
(https://www.pyimagesearch.com/2019/07/15/video-classification-with-keras-and-deep-learning/#comment-525773)

HI Adrian. That was very informative. I had a small question. Can we use the same approach to performance regression. I have a sequence of frames and I would like to use this information to predict a target value, like time remaining. I am trying to do this with CNN and GRU's. However, I am not getting good results with this approach.

Adrian Rosebrock
July 15, 2019 at 1:06 pm
(https://www.pyimagesearch.com/2019/07/15/video-classification-with-keras-and-deep-learning/#comment-525780)

Performance regression as in tests for software regression? If so, how are you using CNNs for that?

Amit Kumar
October 13, 2019 at 1:24 pm
(https://www.pyimagesearch.com/2019/07/15/video-classification-with-keras-and-deep-learning/#comment-563983)

Hi Neeraj,

I am working on a similar problem, can you please help me understand how did you set up the data for that?

Regards,
Amit

David Bonn
July 15, 2019 at 1:20 pm (https://www.pyimagesearch.com/2019/07/15/video-classification-with-keras-and-deep-learning/#comment-525788)

Great post!

I'm really surprised and impressed such a simple approach as rolling prediction averaging works so well.

This post is one of those great posts which forces me to rethink some of what I have been trying to do and rethink some of my assumptions. Thanks.

Adrian Rosebrock
July 16, 2019 at 8:41 am (https://www.pyimagesearch.com/2019/07/15/video-classification-with-keras-and-deep-learning/#comment-525913)

Thanks David!

AASHUTOSH SONI
July 15, 2019 at 3:09 pm
(https://www.pyimagesearch.com/2019/07/15/video-classification-with-keras-and-deep-learning/#comment-525800)

Hey Adrian!

This is Ash. I tried video classification with inceptionV3 as my base model followed by Keras sequential API.I got an accuracy of 93%. Still I am curious about why there is no role of sequential layers like RNN and LSTM to get the relation between each frame.

And 1 more important thing to be noted is, I found the Keras official document in which they said we can convert any image classification model into video classification model just by adding TIMEDISTRIBUTED layer (still I don't know what they want to say)
You can refer to this link: **https://keras.io/getting-started/functional-api-guide/ (https://keras.io/getting-started/functional-api-guide/)**

Looking forward to your reply.
Regards,
Ash

Adrian Rosebrock
July 16, 2019 at 8:41 am
(https://www.pyimagesearch.com/2019/07/15/video-classification-with-keras-and-deep-learning/#comment-525912)

Hey Ash — see the intro to the post. The point of this post is to *not* use RNN and LSTMs and instead discuss how prediction averaging can be sufficient in some cases. They will be covered in a separate guide. TimeDistributed layers, RNNs, and LSTMs are different beasts. I'll cover them later.

Ashish
July 19, 2019 at 7:27 am
(https://www.pyimagesearch.com/2019/07/15/video-classification-with-keras-and-deep-learning/#comment-526327)

Hi Adrian

Great post. Prediction averaging seems like a hack but I suspect this might be one of the better solutions for video classification . Hence I am very much interested in and request for further sequels to this post using TimeDistributed , RNNs and LSTMs. Then we can compare the results also.

Thanks and Keep up the Good work.

Israel (http://None)
July 15, 2019 at 4:24 pm
(https://www.pyimagesearch.com/2019/07/15/video-classification-with-keras-and-deep-learning/#comment-525805)

Hi Dr. Adrián,

Is this included (and reviewed in depth) in Deep Learning for CV? (maybe imagenet bundle)

Adrian Rosebrock
July 16, 2019 at 8:34 am
(https://www.pyimagesearch.com/2019/07/15/video-classification-with-keras-and-deep-learning/#comment-525908)

Are you referring specifically to video classification? Yes, I'm covering video classification and human activity recognition in the 3rd edition of Deep Learning for Computer Vision with Python.

Israel (http://None)
July 20, 2019 at 7:49 pm
(https://www.pyimagesearch.com/2019/07/15/video-classification-with-keras-and-deep-learning/#comment-526542)

Oh! You are amazing! If I've already bought DL4CV, that 3rd Edition (that you've mentioned) is included or I have to pay for it?

Regarding my question: I was referring about dong video classification with CNN + LSTM

Kind Regards.

Adrian Rosebrock
July 25, 2019 at 9:22 am (https://www.pyimagesearch.com/2019/07/15/video-classification-with-keras-and-deep-learning/#comment-527227)

If you already have a copy of DL4CV then you will receive the updated 3rd edition for free when it releases.

If you haven't purchased a copy of DL4CV yet, and would like to, **use this link. (https://www.pyimagesearch.com/deep-learning-computer-vision-python-book/)** Again, you will receive any new updates to the book via email.

Junyang
July 15, 2019 at 10:28 pm
(https://www.pyimagesearch.com/2019/07/15/video-classification-with-keras-and-deep-learning/#comment-525834)

Hi Adrian, thanks for the post. Was wondering, if I were to detect small activities or movement like turning over a book or a card, what implementation would you suggest? I am thinking a naive image classification method wouldn't work because the front and back of the book would be a common feature. That is unless, i train the network to learn, for eg three labels. front, side and back of the book, thoroughly. Normalize the last ~20 frames in a suitable way which detects turning if theres front side and back of books frames detected.

However, I would still like to seek your advice. Thanks!

Adrian Rosebrock
July 16, 2019 at 8:35 am
(https://www.pyimagesearch.com/2019/07/15/video-classification-with-keras-and-deep-learning/#comment-525910)

The approach used in this tutorial would be a good starting point. It may even give you the accuracy you're looking for. Try training a simple model and giving it a try.

Manoj
July 16, 2019 at 1:23 am
(https://www.pyimagesearch.com/2019/07/15/video-classification-with-keras-and-deep-learning/#comment-525848)

Hi Adrian i have one question. Can I use same approach for activity detection in a video ?

Adrian Rosebrock
July 16, 2019 at 8:34 am
(https://www.pyimagesearch.com/2019/07/15/video-classification-with-keras-and-deep-learning/#comment-525909)

Yes, but LSTMs may be more accurate. You should give both a try and see.

Nikhil Chhabra
July 16, 2019 at 2:50 am
(https://www.pyimagesearch.com/2019/07/15/video-classification-with-keras-and-deep-learning/#comment-525858)

Hi,
Thank you for this tutorial. I will try this for Human Activity Recognition.

The challenging thing is to distinguish the process of Sitting and process of Standing up.
Another challenge is to do it in Online fashion.

Adrian Rosebrock
July 16, 2019 at 8:33 am (https://www.pyimagesearch.com/2019/07/15/video-classification-with-keras-and-deep-learning/#comment-525907)

Good luck with it Nikhil!

Ro, sijin
July 16, 2019 at 3:21 am (https://www.pyimagesearch.com/2019/07/15/video-classification-with-keras-and-deep-learning/#comment-525862)

We want to analyze customer behavior using Pose Estimation.
example)
Customer drinking water.
I worry about purchasing products. (Touching glasses, looking at goods for a long time)
How about referring to this tutorial?

Matt
July 16, 2019 at 3:23 am
(https://www.pyimagesearch.com/2019/07/15/video-classification-with-keras-and-deep-learning/#comment-525863)

Why not use a 3D CNN kernel?

Adrian Rosebrock
July 16, 2019 at 8:32 am
(https://www.pyimagesearch.com/2019/07/15/video-classification-with-keras-and-deep-learning/#comment-525905)

See the intro to the post.

Soïta
July 16, 2019 at 3:42 am
(https://www.pyimagesearch.com/2019/07/15/video-classification-with-keras-and-deep-learning/#comment-525866)

Hi Adrian,

Very usefull.

Can we do that with a real time rtsp from ip cameras ?

Adrian Rosebrock
July 16, 2019 at 8:33 am
(https://www.pyimagesearch.com/2019/07/15/video-classification-with-keras-and-deep-learning/#comment-525906)

Yes. I would recommend using **ImageZMQ. (https://www.pyimagesearch.com/2019/04/15/live-video-streaming-over-network-with-opencv-and-imagezmq/)**

Toshio Futami
July 16, 2019 at 3:44 am
(https://www.pyimagesearch.com/2019/07/15/video-classification-with-keras-and-deep-learning/#comment-525868)

I succeeded correct labeling for only sample_clips which you prepared.
I tried this prediction for boxing video which I added to sample_clips folder. But it mistakes as tennis or lifting. Same situation for fencing!
Please let me know if you have any improvement way.

Adrian Rosebrock
July 16, 2019 at 8:32 am
(https://www.pyimagesearch.com/2019/07/15/video-classification-with-keras-and-deep-learning/#comment-

525904)

The model in this post was only trained on “football”, “tennis”, and “weight_lifting” classes. If you wanted to recognize fencing or boxing you would need to include those classes in the training.

Toshio Futami
July 17, 2019 at 10:49 pm (https://www.pyimagesearch.com/2019/07/15/video-classification-with-keras-and-deep-learning/#comment-526160)

Thank you for your advice.

subbu
July 16, 2019 at 4:52 am (https://www.pyimagesearch.com/2019/07/15/video-classification-with-keras-and-deep-learning/#comment-525878)

Hi Adrian, Thanks for this article. I had one question , if i need to identify a activity in video and extract only those frames what should be the approach(will LSTM work?). Example: if i want to extract only Goals from a football video ,how can this be achieved?

Adrian Rosebrock
July 16, 2019 at 8:31 am
(https://www.pyimagesearch.com/2019/07/15/video-classification-with-keras-and-deep-learning/#comment-525903)

> Detecting goals themselves could combine a few different methods. You could use sensor fusion and monitor the ball itself (most accurate). You could mount a camera on top of the goal post and then monitor the goal line (also accurate). Trying to extract just goals from raw footage would be much more challenging though.

Mohammed Moussa
July 16, 2019 at 5:19 am
(https://www.pyimagesearch.com/2019/07/15/video-classification-with-keras-and-deep-learning/#comment-525880)

Hi Adrian, thanks for this interesting tutorial. I have a theoretical question related to the concept of this tutorial, which is :

If I want to further classify the same activity by interpreting the reason why such activity is taking place, such as this example:

“I have 2 videos in each people running (activity = run), in the first one, they’re running because of competing in a marathon race, while in the other they are running away from danger.”

Do you think by training a CNN with enough dataset would be sufficient to build a model that is capable to classify with reasoning? what is your theoretical pipeline for such scenario?

Adrian Rosebrock
July 16, 2019 at 8:30 am (https://www.pyimagesearch.com/2019/07/15/video-classification-with-keras-and-deep-learning/#comment-525902)

It's honestly hard to say without running experiments to first verify. Visually there are distinct differences between someone running for pleasure and sport versus running to escape and fear. With the proper training data it may be possible but I would run some initial experiments and let the empirical results guide you.

Mohammed Moussa
July 16, 2019 at 10:18 am (https://www.pyimagesearch.com/2019/07/15/video-classification-with-keras-and-deep-learning/#comment-525929)

That would be great, thank you very much for the answer and the efforts 🙂

Walid
July 16, 2019 at 9:06 am
(https://www.pyimagesearch.com/2019/07/15/video-classification-with-keras-and-deep-learning/#comment-525918)

Great post as usual
I applied your approach to the UCF101 dataset and it worked very good as well
Here is a demo for the video classification
https://www.youtube.com/watch?v=SwaX6L7zpNs&t=8s (https://www.youtube.com/watch?v=SwaX6L7zpNs&t=8s)

Adrian Rosebrock
July 18, 2019 at 8:40 am
(https://www.pyimagesearch.com/2019/07/15/video-classification-with-keras-and-deep-learning/#comment-526210)

Thanks for sharing Walid, great job!

I Sharma
August 29, 2019 at 6:25 am
(https://www.pyimagesearch.com/2019/07/15/video-classification-with-keras-and-deep-learning/#comment-545793)

Can you please share your code.

Thanks

tool (http://no)
July 16, 2019 at 9:06 am
(https://www.pyimagesearch.com/2019/07/15/video-classification-with-keras-and-deep-learning/#comment-525919)

Hi, Adrian awesome guide, one question how can i train whit my own video dataset? by the way any source for 3D-CNN ?

Again great and wonderful work your site is awesome

Adrian Rosebrock
July 18, 2019 at 8:39 am
(https://www.pyimagesearch.com/2019/07/15/video-classification-with-keras-and-deep-learning/#comment-526208)

I'll be covering working with video sequences directly inside the CNN in a future tutorial.

Vismaya Prasannakumar
March 19, 2020 at 6:45 am
(https://www.pyimagesearch.com/2019/07/15/video-classification-with-keras-and-deep-learning/#comment-766889)

Hi Andrian,

Thanks for the awesome tutorial.
Any updates on training the model directly using video files ?
How can I train a video dataset using the CNN(like UCF-101 or personal data)

Thanks,
Vismaya

Adrian Rosebrock
March 19, 2020 at 9:41 am (https://www.pyimagesearch.com/2019/07/15/video-classification-with-keras-and-deep-learning/#comment-766914)

I haven't written that tutorial yet, I've been too busy with COVID-related work.

Jorge
July 16, 2019 at 9:16 am (https://www.pyimagesearch.com/2019/07/15/video-classification-with-keras-and-deep-learning/#comment-525922)

Hi, Adrian. Would it be possible to train a video classifier similar to the one you propose in this article by means of transfer learning to use it in preventive safety systems? For example, detecting a person in a climbing attitude of a private property

wall from security cameras, or in a suspicious attitude in front of the same wall, differentiating it from the person who is simply in the place in a passive attitude or ringing the bell. (I suppose we would have to generate datasets representative of the different situations) Maybe there are already such security systems based on LSTM / RNN? What approach would you suggest?

Jorge, from Argentina

Adrian Rosebrock
July 18, 2019 at 8:39 am
(https://www.pyimagesearch.com/2019/07/15/video-classification-with-keras-and-deep-learning/#comment-526207)

This model was actually trained via transfer learning/fine-tuning. If you already have example image/video data of your security application I would suggest training the model and seeing what results you get (it's hard to definitively say without seeing your data). Use this method as a first approach to obtain a baseline accuracy, then move on to RNNs/LSTMs.

Sanchit Jain
July 17, 2019 at 1:44 am
(https://www.pyimagesearch.com/2019/07/15/video-classification-with-keras-and-deep-learning/#comment-526027)

Hi Adrian,

I executed this code in my laptop but not able to achieve the speed , I mean the video when it runs it's quite slow as compared to what shown in the post , is it depending on the laptop configuration or something else? kindly let me know.

My processor is intel corei5 8th gen, 16GB RAM .

Thank you.

Adrian Rosebrock
July 18, 2019 at 8:37 am
(https://www.pyimagesearch.com/2019/07/15/video-classification-with-keras-and-deep-learning/#comment-526206)

For faster training, for this method to run in real-time, you would need to use a GPU.

Adrian Rosebrock
August 16, 2019 at 5:22 am
(https://www.pyimagesearch.com/2019/07/15/video-classification-with-keras-and-deep-learning/#comment-539168)

The images are downloaded when do a "git clone" of the repo.

khan
July 17, 2019 at 9:01 am
(https://www.pyimagesearch.com/2019/07/15/video-classification-with-keras-and-deep-learning/#comment-526061)

Hi adrian,

do u think if we have 100 clips, 50 each of two different classes . e.g, classification of different soccer kicks(corner kicks, freekicks) each 8 to 10 secs. if we train the model ll it be able distinguish between the kicks.

Adrian Rosebrock
July 18, 2019 at 8:37 am
(https://www.pyimagesearch.com/2019/07/15/video-classification-with-keras-and-deep-learning/#comment-526205)

> It's hard to say without seeing the data first. Try it and see! Let your empirical results guide you.

Aiden Ralph
July 17, 2019 at 10:11 am
(https://www.pyimagesearch.com/2019/07/15/video-classification-with-keras-and-deep-learning/#comment-526076)

Hi Adrian,

Very cool post, very detailed. Sort of eliminates the false positives from a video.
Just wondering a more simpler version of that, in a normal object detection or image classification (like your famous santa claus detector tutorial), can you make the python program do something (sound an alarm etc…) when it detects say 3 frames in a row (or quick succession).
I sometimes get false-positives in one frame of a video for example and no matter how much tuning or different ways of making classifiers or trained models I can't solve that one false positive, but Im sure it can be pythoned out. Do you know what I mean?

Adrian Rosebrock
July 18, 2019 at 8:37 am
(https://www.pyimagesearch.com/2019/07/15/video-classification-with-keras-and-deep-learning/#comment-526204)

Yep! Checking for N consecutive frames with a positive, identical prediction is a fairly common technique. It's a nice, easy "hack" that can sometimes work.

Aqsa
July 17, 2019 at 2:27 pm
(https://www.pyimagesearch.com/2019/07/15/video-classification-with-keras-and-deep-learning/#comment-526110)

this is very interesting tutorial , i am now to deep learning so my question is this source code will run in google colab , if yes then how 🙂

Adrian Rosebrock
July 18, 2019 at 8:36 am (https://www.pyimagesearch.com/2019/07/15/video-classification-with-keras-and-deep-learning/#comment-526203)

Yes, but you would need to upload the data to your Google Drive account. I don't have any tutorials on using Google Colab but I know there is interest in it so I may write one in the future.

Md. Imrul Hasan
March 8, 2020 at 1:42 pm (https://www.pyimagesearch.com/2019/07/15/video-classification-with-keras-and-deep-learning/#comment-765815)

Yes. You can write in colab. I ran the full code both in colab and jupyter notebook. I trained the model in colab. And ran the testing part in Jupyter. Because in colab I was unable to see output as video instead in frames.

Muhammad Andyk Maulana (http://muhammadandykmaulana.github.io)

July 17, 2019 at 11:06 pm
(https://www.pyimagesearch.com/2019/07/15/video-classification-with-keras-and-deep-learning/#comment-526161)

Hi Adrian, Thanks for wirklich interessant tutorial. Did you try image/video classification for capturing coal which is running on conveyor? If so, i'm obsfucated to find image datasets for coal.

Thanks for reading my comment.

Regards,
Andyk

Adrian Rosebrock
July 18, 2019 at 8:35 am
(https://www.pyimagesearch.com/2019/07/15/video-classification-with-keras-and-deep-learning/#comment-526202)

Sorry, I do not have any tutorials for that. If you have any example images/video you're working with let me know and I may consider writing a tutorial for it in the future.

Phil
July 18, 2019 at 10:21 am
(https://www.pyimagesearch.com/2019/07/15/video-classification-with-keras-and-deep-learning/#comment-526213)

What is the point of image mean subtraction here?

Adrian Rosebrock
July 25, 2019 at 9:25 am
(https://www.pyimagesearch.com/2019/07/15/video-classification-with-keras-and-deep-learning/#comment-527233)

Mean subtraction is a form of normalization/scaling. It's covered in-depth inside **Deep Learning for Computer Vision with Python. (https://www.pyimagesearch.com/deep-learning-computer-vision-python-book/)**

Antonio Lyu
July 19, 2019 at 5:11 am
(https://www.pyimagesearch.com/2019/07/15/video-classification-with-keras-and-deep-learning/#comment-526306)

Hi Adrian,

Very useful，thanks

In this example we noticed that we only used very little training data. what if we have one million or more training data? We don't have that much memory, and the way we read images today useing Opencv is too slow. So I am very much looking

forward to find a tutorial to help me, although I know that there are some good data formats like HDF5 TFrecords, but I don't know how to use them in projects to train our network like today.

Best Regards,

Lyu

Adrian Rosebrock
July 25, 2019 at 9:24 am
(https://www.pyimagesearch.com/2019/07/15/video-classification-with-keras-and-deep-learning/#comment-527231)

Great question, Lyu. I cover how to (1) build an image dataset using HDF5 and (2) train a CNN using the HDF5 dataset inside my book, **Deep Learning for Computer Vision with Python. (https://www.pyimagesearch.com/deep-learning-computer-vision-python-book/)** I would definitely encourage you to start there.

Akash
July 19, 2019 at 6:09 am
(https://www.pyimagesearch.com/2019/07/15/video-classification-with-keras-and-deep-learning/#comment-526316)

Hi Adrian,

Comparing to Breast Cancer identification, in the ImageDataGenerator, you used rescale in that one. In this, there is no rescale between 0 to 1. Why that so? Isn't it advised to normalize between 0 & 1?

Or there is difference due to the use of ImageNet classification model? Here, it's mean substraction too that is specifically included due to ImageNet models.

Thanks for the great post.

Adrian Rosebrock
July 25, 2019 at 9:23 am
(https://www.pyimagesearch.com/2019/07/15/video-classification-with-keras-and-deep-learning/#comment-527229)

We don't do [0, 1] normalization here because we are performing mean subtraction.

deni
July 19, 2019 at 6:17 am
(https://www.pyimagesearch.com/2019/07/15/video-classification-with-keras-and-deep-learning/#comment-526318)

hello andrian. thank you for the tutorial, what a coincidence, I was looking for a way to classify the video this week, and you came up with this great tutorial :).

By the way, I tried to train all existing classes (20 classes). if I use 1 gpu I get 70% accuracy. however, when I use multiple gpu (2 gpu) I get 84% accuracy. why are the results so different?

Adrian Rosebrock
July 25, 2019 at 9:23 am
(https://www.pyimagesearch.com/2019/07/15/video-classification-with-keras-and-deep-learning/#comment-527228)

Try re-running the experiment multiple times on a single GPU. It sounds like your dataset is small or your hyperparameters are tuned properly, leading to very different results.

deni
July 30, 2019 at 4:12 am
(https://www.pyimagesearch.com/2019/07/15/video-classification-with-keras-and-deep-learning/#comment-528444)

does that mean, the number of gpu used affects accuracy?
and accuracy can be different if run many times?

nb: in my experiment, I trained all classes (20) , whereas in the example above only 3 classes were trained.

Adrian Rosebrock
August 7, 2019 at 12:54 pm (https://www.pyimagesearch.com/2019/07/15/video-classification-with-keras-and-deep-learning/#comment-532754)

No, the number of GPUs will not affect accuracy. Instead, it's the nature of stochastic algorithms on small datasets.

Shahid khan
July 20, 2019 at 12:51 am (https://www.pyimagesearch.com/2019/07/15/video-classification-with-keras-and-deep-learning/#comment-526418)

sir please make a tutorial about action detection.

Vladimir Ovsiannikov
July 20, 2019 at 11:51 pm (https://www.pyimagesearch.com/2019/07/15/video-classification-with-keras-and-deep-learning/#comment-526568)

Hi Adrian! Thank you for the lesson. In this example, the model.predict () method is very slow. Is it possible to somehow speed up this part of the code, for example, apply multiprocessing?

Adrian Rosebrock
July 25, 2019 at 9:21 am
(https://www.pyimagesearch.com/2019/07/15/video-classification-with-keras-and-deep-learning/#comment-527226)

Are you using a CPU? If so, try using a GPU instead.

Gagandeep
July 22, 2019 at 3:36 am
(https://www.pyimagesearch.com/2019/07/15/video-classification-with-keras-and-deep-learning/#comment-526714)

Again thanks for the one more great tutorial, I need some expertise suggestion on my current deep learning project. Application is capsule defect inspection using deep leaning.

This is my first deep learning project and I have very less experience in deep learning field, so I would like to ask some few questions mentioned below:

1 which neural n/w could be a best choice to detect the 10×10 defect in the input image. (Re-tuning can be done for caffe and tensorflow neural networks with the available training dataset with image resolution 512×320)

2: Is it possible to detect the min. 10×10 defected area using the 512×320 datasets capsule images?

FYI Inference engine will run on xilinx 7EV device. which convert this floating point n/w to interger8,

your valuable suggestions will be greatly appreciated.

Adrian Rosebrock
July 25, 2019 at 9:21 am (https://www.pyimagesearch.com/2019/07/15/video-classification-with-keras-and-deep-learning/#comment-527225)

I would suggest taking a look at **Deep Learning for Computer Vision with Python. (https://www.pyimagesearch.com/deep-learning-computer-vision-python-book/)** It covers my tips, suggestions, and best practices when training your own networks.

Joker
July 22, 2019 at 11:54 am
(https://www.pyimagesearch.com/2019/07/15/video-classification-with-keras-and-deep-learning/#comment-526755)

Hi Adrian, Thanks for this interesting tutorial. I have a question about the context of CNN and LSTM. I have trained a CNN network for image classification. However, I would like to combine it with LSTM for visualizing the attention weights. So, I extracted the features from the CNN to put it into LSTM. However, I am stuck at the concept of combinating the CNN with LSTM.

– Do I need to train the whole network again? Or just training the LSTM part is fine?
– Can I just train the LSTM on image sequences based on classes (for e.g. 1 class has around 300 images) and do predictions later on extracted video frames?

I hope you can help me while I struggle with the context of understanding the combination of this.

Adrian Rosebrock
July 25, 2019 at 9:20 am
(https://www.pyimagesearch.com/2019/07/15/video-classification-with-keras-and-deep-learning/#comment-527224)

> I don't have any tutorials on LSTMs yet but I'll keep your question in mind for when I write a tutorial on it. Thanks for the suggestions/questions!

Gautam Kumar (http://www.nitrkl.ac.in)
July 24, 2019 at 4:33 am
(https://www.pyimagesearch.com/2019/07/15/video-classification-with-keras-and-deep-learning/#comment-527061)

Another excellent tutorial by you. Thanks for sharing code. I was looking for a similar idea and thought to mail you regarding this last month but I didn't. Anyways, I have trained the model using your code. I cropped two video clips say, football and chess, and merged these clips into a single video to test model performance. Between these two video clips there is a scene of around 15 second were person is neither playing football nor chess still model predict those 15 second scene as football. I want to modify your code for my academic purpose in such a way that system should predict only different types of game and if there are some scene in video on which model is not trained with (like 15 second scene), it shoud not show any output on output video screen. Any idea, suggestion or reference would be appreciated. Thanks once again for sharing knowledge.

Adrian Rosebrock
July 25, 2019 at 9:20 am
(https://www.pyimagesearch.com/2019/07/15/video-classification-with-keras-and-deep-learning/#comment-527223)

Basically, you want to include an “ignore” class in your training (sometimes called a “background” class for object detection and instance segmentation). Fill this class with images that are NOT sports. Then, train the model on your normal sports classes and the background/ignore class.

Nel
July 24, 2019 at 9:29 am
(https://www.pyimagesearch.com/2019/07/15/video-classification-with-keras-and-deep-learning/#comment-527102)

Hello Adrian,

Thanks for this article. I need your advice about my project. I am going to classify 3 different level of depression from video recordings. I have a loop through all images of a video and used fine-tuned vggface to extract features of all images and append them to make a sequence. Then I need to feed each sequence to the LSTM for classification. But the validation

accuracy remains constant. Can I use this approach instead of LSTM? And I can not use fit.genarator because I can not separate different labels.

Thanks,

Adrian Rosebrock
July 25, 2019 at 9:19 am
(https://www.pyimagesearch.com/2019/07/15/video-classification-with-keras-and-deep-learning/#comment-527222)

I would suggest you try it and compare it to your LSTM, that way you have a baseline accuracy to work from.

tool
July 24, 2019 at 11:38 am
(https://www.pyimagesearch.com/2019/07/15/video-classification-with-keras-and-deep-learning/#comment-527117)

I have a question what should i change from the code to work with my own dataset?

Adrian Rosebrock
July 25, 2019 at 9:13 am
(https://www.pyimagesearch.com/2019/07/15/video-classification-with-keras-and-deep-learning/#comment-527215)

Do you want to train your own custom CNN? Or do you want to apply a pre-trained CNN to image classification?

tool
July 24, 2019 at 12:28 pm
(https://www.pyimagesearch.com/2019/07/15/video-classification-with-keras-and-deep-learning/#comment-527120)

Another question sr. Why tthe queue of 128? What it means i terms of the classification ? somethiong happens if my videos are shorter?

Adrian Rosebrock
July 25, 2019 at 9:13 am
(https://www.pyimagesearch.com/2019/07/15/video-classification-with-keras-and-deep-learning/#comment-527214)

I tuned the 128 value experimentally. It worked well for the video classification.

tool
July 25, 2019 at 10:31 pm
(https://www.pyimagesearch.com/2019/07/15/video-classification-with-keras-and-deep-learning/#comment-527387)

But what's the meaning of that 128? Is there any reference to that?

Thanks in advance

Adrian Rosebrock
August 7, 2019 at 1:11 pm (https://www.pyimagesearch.com/2019/07/15/video-classification-with-keras-and-deep-learning/#comment-532788)

It's the number of frames in the rolling average.

Nel
July 25, 2019 at 11:04 am (https://www.pyimagesearch.com/2019/07/15/video-classification-with-keras-and-deep-learning/#comment-527313)

Thanks for your answer. By the way my model doesn't work well. My validation accuracy remains constant at 40% and training accuracy goes up and down! Can you guess what is wrong? Cause I have checked everything and they were fine.

Thanks

Adrian Rosebrock
August 7, 2019 at 1:23 pm
(https://www.pyimagesearch.com/2019/07/15/video-classification-with-keras-and-deep-learning/#comment-532797)

Hey Nel — are you using the dataset I used for this guide? Or a different dataset?

For what it's worth, I cover my tips, suggestions, and best practices to training your own custom deep learning models inside **Deep Learning for Computer Vision with Python. (https://www.pyimagesearch.com/deep-learning-computer-vision-python-book/)** I would definitely suggest starting there.

Dima
July 25, 2019 at 11:06 am
(https://www.pyimagesearch.com/2019/07/15/video-classification-with-keras-and-deep-learning/#comment-527314)

Hi, i download source code, but folder model and was empty, how i can download model?

Adrian Rosebrock
August 7, 2019 at 1:23 pm
(https://www.pyimagesearch.com/2019/07/15/video-classification-with-keras-and-deep-learning/#comment-532796)

You need to run the Python scripts included in the download to generate the model files.

Jhonatan Silva Souza
July 29, 2019 at 12:35 am (https://www.pyimagesearch.com/2019/07/15/video-classification-with-keras-and-deep-learning/#comment-528008)

I download the source code, but theres nothing in the model folder

---

Adrian Rosebrock
August 7, 2019 at 1:00 pm (https://www.pyimagesearch.com/2019/07/15/video-classification-with-keras-and-deep-learning/#comment-532766)

Run the Python scripts to generate the output model and pickle file.

Nakul
July 29, 2019 at 11:44 am (https://www.pyimagesearch.com/2019/07/15/video-classification-with-keras-and-deep-learning/#comment-528213)

Hello Adrian great post can you please tell me if this code works in windows machine?
I also tried to run it in MAC but I'm facing issue with train.py it gives me error of SSL certificate error while fetching resnet50 model. Please help.

Adrian Rosebrock
August 7, 2019 at 12:58 pm
(https://www.pyimagesearch.com/2019/07/15/video-classification-with-keras-and-deep-learning/#comment-532759)

That sounds like a DNS or network issue of some sort on your Mac. I would double-check your network settings.

Siddhesh
July 29, 2019 at 12:23 pm
(https://www.pyimagesearch.com/2019/07/15/video-classification-with-keras-and-deep-learning/#comment-528222)

Hi Thanks for the great post. I am working on project where I want to classify an event in a football match for eg freekick, goal, corner, redcard/yellowcard. What you can suggest for me to do in this case. The data is in the form of video snippet each representing the event. I want to classify the whole video as an event.

Adrian Rosebrock
August 7, 2019 at 12:58 pm
(https://www.pyimagesearch.com/2019/07/15/video-classification-with-keras-and-deep-learning/#comment-532758)

What methods have you tried thus far? Have you tried using the techniques in this post as a starting point?

khan
July 30, 2019 at 11:39 am
(https://www.pyimagesearch.com/2019/07/15/video-classification-with-keras-and-deep-learning/#comment-528544)

hi adrian, i am using two classes with forty clips each mostly system predicts two classes e.g for cornerkick in soccer it initially predicts it as corrnerkick then in the clip it predicts it as freekick.

Adrian Rosebrock
August 7, 2019 at 12:52 pm
(https://www.pyimagesearch.com/2019/07/15/video-classification-with-keras-and-deep-learning/#comment-532750)

Try adjusting the “–size” command line argument. Otherwise, you may need a more advanced method, such as RNNs or LSTMs.

thekhieu
July 31, 2019 at 3:06 am
(https://www.pyimagesearch.com/2019/07/15/video-classification-with-keras-and-deep-learning/#comment-528850)

i have downloaded keras-video-classification.zip but model folder is empty? Can you share it to me? thank you

---

Adrian Rosebrock
August 7, 2019 at 12:47 pm
(https://www.pyimagesearch.com/2019/07/15/video-classification-with-keras-and-deep-learning/#comment-532742)

Run the Python scripts to generate the model and pickle files.

m
July 31, 2019 at 7:46 am
(https://www.pyimagesearch.com/2019/07/15/video-classification-with-keras-and-deep-learning/#comment-528945)

Hi Adrian,

Thank you for your great tutorials!

One issue though, I went to the repo you referenced here & I couldn't find the activity.model file which you mentioned! The owner seems to have changed the location of his data & models but even with the new location I can only find .pth files?! Could you maybe share a link of the model & .pkl files if you have it so we can follow with your tutorial?

thanks

Adrian Rosebrock
August 7, 2019 at 12:46 pm (https://www.pyimagesearch.com/2019/07/15/video-classification-with-keras-and-deep-learning/#comment-532740)

I think you're confusing the code download and the dataset download:

1. I provide a link to the GitHub repo for the dataset (make sure you download it)
2. Then, use the "Downloads" section of this tutorial to download the code
3. Run the Python scripts to generate the model and pickle files

---

m
August 15, 2019 at 1:35 pm (https://www.pyimagesearch.com/2019/07/15/video-classification-with-keras-and-deep-

learning/#comment-538582)

Thanks for the response Adrian! I was indeed confused!
So I've used my own dataset (mainly from googel)for some human activities like reading & writing & I notice that I barely reach 50% accuracy with 0.6 as min loss.
I was wondering if small (around 300 to 700 per category) &repeated images would affect the results that much? and if there any way to enhance the results?

Adrian Rosebrock
August 16, 2019 at 5:29 am (https://www.pyimagesearch.com/2019/07/15/video-classification-with-keras-and-deep-learning/#comment-539183)

It sounds like you need to tune your hyperparamters to the model. I would suggest you refer to **Deep Learning for Computer Vision with Python (https://www.pyimagesearch.com/deep-learning-computer-vision-python-book/)** where I teach you how to tune your hyperparameters and obtain better model performance.

m
August 26, 2019 at 6:55 am (https://www.pyimagesearch.com/2019/07/15/video-classification-with-keras-and-deep-learning/#comment-545280)

Thanks Adrian. I got one of your books & will make sure to get this one too!

Another question, is there any required size for the images in the dataset?
I'm trying to run it on my own 2 classes dataset but I keep getting this error
"Error when checking target: expected dense_2 to have shape (2,) but got array with shape (1,)"

I run this before & it worked fine. I've kept everything the same except that I noticed the Resnet has been re-downloaded saying the weights has been updated or something
..any idea why I'm getting this error?

Adrian Rosebrock
September 5, 2019 at 10:00 am (https://www.pyimagesearch.com/2019/07/15/video-classification-with-keras-and-deep-learning/#comment-547066)

Double-check:

1. Your input file paths
2. Your label parsing

It sounds like your dataset does not follow the "/dataset_name/class_label_name/image.jpg" directory structure. Make sure it does. Secondly, double-check your label parsing output. It sounds like the labels aren't being properly parsed.

khan
July 31, 2019 at 9:57 am (https://www.pyimagesearch.com/2019/07/15/video-classification-with-keras-and-deep-learning/#comment-528981)

another question, how can we compute the correctness of the output e.g in a video how can we compute 80% of the clip is correctly classfied for one class, where as 15 % is classfied for another class.

Adrian Rosebrock
August 7, 2019 at 12:45 pm
(https://www.pyimagesearch.com/2019/07/15/video-classification-with-keras-and-deep-learning/#comment-532739)

1. Loop over all frames
2. Make predictions on all frames
3. Count the number of times the prediction was correct
4. Divide by total number of frames

That will give you the total percentage of correctly classified frames.

m
August 5, 2019 at 5:30 am
(https://www.pyimagesearch.com/2019/07/15/video-classification-with-keras-and-deep-learning/#comment-531203)

Hello Adrian,
I can’t find the activity and the pickle files from the link you mentioned.

Any other way to get these two files?

Adrian Rosebrock
August 7, 2019 at 12:25 pm
(https://www.pyimagesearch.com/2019/07/15/video-classification-with-keras-and-deep-learning/#comment-532703)

Once you train the model the pickle files will be saved to disk.

Arian
August 5, 2019 at 7:36 pm
(https://www.pyimagesearch.com/2019/07/15/video-classification-with-keras-and-deep-learning/#comment-531624)

Btw, I loved your example. I'm new to DL. Can you please post a tutorial on how to train a LSTM/RNN on a video?

Adrian Rosebrock
August 7, 2019 at 12:22 pm
(https://www.pyimagesearch.com/2019/07/15/video-classification-with-keras-and-deep-learning/#comment-532698)

Yes, I'll be doing guides on LSTMs and RNNs in the future.

Naveen
August 8, 2019 at 5:37 am
(https://www.pyimagesearch.com/2019/07/15/video-classification-with-keras-and-deep-learning/#comment-533204)

Hello Adrian,

Thanks for the tutorial, it's really helpful.
I'm working on a project where I have to detect a road accidents in a video. What would be your suggestion.

Thanks

Haris
August 11, 2019 at 12:38 pm
(https://www.pyimagesearch.com/2019/07/15/video-classification-with-keras-and-deep-learning/#comment-535069)

Hy adrian! i have learned so much under your teaching. God bless you ……. i am working with dynamic hand gesture recognition system (swipe right,left,up,down etc..) i tried this tutorial for this problem but didn't find results. Is simple 3dcnn only will be helpfull for this problem.

Adrian Rosebrock
August 16, 2019 at 5:51 am
(https://www.pyimagesearch.com/2019/07/15/video-classification-with-keras-and-deep-learning/#comment-539228)

Thanks Haris, I'm glad you enjoyed the tutorial. I may cover some more advanced video classification methods in the future, stay tuned!

Walid
August 13, 2019 at 5:52 pm
(https://www.pyimagesearch.com/2019/07/15/video-classification-with-keras-and-deep-learning/#comment-537091)

Hi Adrian
The model is great and worked as exoected but I would really need it in TF.
Can you please have a post to convert keras model to TensorFlow one?

Thanks a lot

Adrian Rosebrock
August 16, 2019 at 5:38 am
(https://www.pyimagesearch.com/2019/07/15/video-classification-with-keras-and-deep-learning/#comment-

539200)

Thanks for the suggestion. I may cover it in the future but I cannot guarantee if/when that may be.

yahuyang
August 16, 2019 at 3:16 am (https://www.pyimagesearch.com/2019/07/15/video-classification-with-keras-and-deep-learning/#comment-539090)

Hello Adrian,
I don't know if the data in the data folder under the Sports-Type-Classifier folder is downloaded from the Internet using the sports_classifier.ipynb code or is the training data downloaded directly from the Internet?

Adrian Rosebrock
August 16, 2019 at 5:22 am (https://www.pyimagesearch.com/2019/07/15/video-classification-with-keras-and-deep-learning/#comment-539169)

The images are downloaded when do a "git clone" of the repo.

Jackson Sandland
August 26, 2019 at 7:50 pm
(https://www.pyimagesearch.com/2019/07/15/video-classification-with-keras-and-deep-learning/#comment-545377)

Hi Adrian,

As evidenced in the comments, a lot of people are confused about the “missing” files from the model directory.
This is because you imply the files are already included before you’ve trained the data when you say: “Our classifier files are in the model/ directory. Included are activity.model (the trained Keras model) and lb.pickle (our label binarizer).”

To stop the confusion, you should reword this to say that the files won’t exist until after you have trained the model.

Adrian Rosebrock
September 5, 2019 at 9:59 am
(https://www.pyimagesearch.com/2019/07/15/video-classification-with-keras-and-deep-learning/#comment-547065)

Thanks Jackson.

Raju
September 5, 2019 at 1:09 am
(https://www.pyimagesearch.com/2019/07/15/video-classification-with-keras-and-deep-learning/#comment-546958)

Hi Adrian,
Using CNN or any other algorithm is it possible to classify the number of sixers in cricket match. Mention any other algorithm suitable for this classfication.

Anh Tuấn Hoàng
September 10, 2019 at 11:05 pm
(https://www.pyimagesearch.com/2019/07/15/video-classification-with-keras-and-deep-learning/#comment-549799)

I think not good if we classify each frame to summarize video. In future, will you write about the-state-of-art algorithms? Can you give me some keyword about it? (algorithms)

Adrian Rosebrock
September 12, 2019 at 11:14 am
(https://www.pyimagesearch.com/2019/07/15/video-classification-with-keras-and-deep-learning/#comment-550293)

Yes, I'll be covering RNNs and LSTMs at a later date.

Kathan Sheth
October 6, 2019 at 11:09 pm
(https://www.pyimagesearch.com/2019/07/15/video-classification-with-keras-and-deep-learning/#comment-559494)

Hi Adrian,
I have a question. Will this approach work if my test video includes multiple labels? What if first five seconds labeled as weight_lifting and next five seconds labeled as swimming? In general, what if my train and test dataset has videos where individual videos contain multiple classes?

Vitor Luiz (https://www.linkedin.com/in/vitorlui/)
October 10, 2019 at 8:38 am
(https://www.pyimagesearch.com/2019/07/15/video-classification-with-keras-and-deep-learning/#comment-563631)

Looking forward for the LSTM tutorial, I hope a chapter of HAR in the next edition of deep learning book 😉 Thanks Adrian, great explanation as always!

Adrian Rosebrock
October 10, 2019 at 10:06 am
(https://www.pyimagesearch.com/2019/07/15/video-classification-with-keras-and-deep-learning/#comment-563639)

Hey Victor — I'll be doing some LSTM and HAR content soon 🙂

ALTAF HUSSAIN
October 13, 2019 at 7:17 am (https://www.pyimagesearch.com/2019/07/15/video-classification-with-keras-and-deep-learning/#comment-563949)

Thanks Adrian, i have a question please guide us how to load video dataset instead of images. your imutils is only load images please add functionalities for loading video dataset also. thanks

---

Adrian Rosebrock
October 17, 2019 at 7:58 am (https://www.pyimagesearch.com/2019/07/15/video-classification-with-keras-and-deep-learning/#comment-564810)

A video is just a series of individual images. Loop over the frames of your video and save them to disk. From there you can apply the code used in this tutorial.

Ramy Hussein

October 18, 2019 at 1:26 am
(https://www.pyimagesearch.com/2019/07/15/video-classification-with-keras-and-deep-learning/#comment-565046)

Hi Adrian,
The data is not available on Github anymore. Has the author uploaded it to another repo?
Thanks!

okorie emmanuel
November 6, 2019 at 6:19 am
(https://www.pyimagesearch.com/2019/07/15/video-classification-with-keras-and-deep-learning/#comment-570125)

Thanks, Adrian for this instructive tutorial. My question is how can I do video classification where my dataset is videos and also feed in a video for prediction

---

Adrian Rosebrock
November 7, 2019 at 10:09 am
(https://www.pyimagesearch.com/2019/07/15/video-classification-with-keras-and-deep-learning/#comment-570718)

Refer to the comments section of the post as I've answered that question. Basically you create your image dataset from the video by sampling every N-th frame

from the video. You don't want to use every frame as that would result in a huge explosion of data, and not to mention, frames that are very similar.

okorie emmanuel
November 13, 2019 at 1:38 pm (https://www.pyimagesearch.com/2019/07/15/video-classification-with-keras-and-deep-learning/#comment-572913)

Thanks for your response. I'm just wondering how i can achieve this assumimg i have a dataset for instance 100 different sport activites with each spot activity having 10 to 15 vidoes each. Do i need to create image datasets for each of the 10 to 15 videos for a spot activity and for the 100 different sport activities, how will one store and arrange the images and wouldn't images be too difficult to arrange and store.

Adrian Rosebrock
November 14, 2019 at 9:12 am (https://www.pyimagesearch.com/2019/07/15/video-classification-with-keras-and-deep-learning/#comment-573208)

Have you taken a look at **Deep Learning for Computer Vision with Python (https://www.pyimagesearch.com/deep-**

learning-computer-vision-python-book/)? That book covers how to build your own image datasets and organize them properly on disk. I would suggest starting there.

okorie emmanuel
November 17, 2019 at 11:05 pm (https://www.pyimagesearch.com/2019/07/15/video-classification-with-keras-and-deep-learning/#comment-574813)

ok. Thanks

Ameya
November 12, 2019 at 2:58 pm (https://www.pyimagesearch.com/2019/07/15/video-classification-with-keras-and-deep-learning/#comment-572491)

Hi Adrian, amazing work!
I have a question. I tried scaling the model to work with 4 sports. It successfully loaded the images for the new sport (badminton) but then it does not load the images for the remaining of the sports and that is why the ‘cv2.cvtColor()’ function gives an error. I dont exactly know why the images are not been loaded properly. Can you please help me on this?

Adrian Rosebrock
November 14, 2019 at 9:13 am
(https://www.pyimagesearch.com/2019/07/15/video-classification-with-keras-and-deep-learning/#comment-573210)

It's hard to say without seeing the exact error message. I would suggest you read **Deep Learning for Computer Vision with Python (https://www.pyimagesearch.com/deep-learning-computer-vision-python-book/)** which includes my tips, suggestions, and best practices when building your own custom image datasets and training networks on them.

Yousmoty
November 13, 2019 at 7:27 am
(https://www.pyimagesearch.com/2019/07/15/video-classification-with-keras-and-deep-learning/#comment-572717)

Thanks for the tutorial Adrian.
The data is still in the repo, just in a different link. Needs to be downloaded and extracted and placed in the correct directory.

I was receiving a key error for the plot. I had to change val_acc to val_accuracy and acc to accuracy. Any ideas why?
Thanks again for the tutorial!

Adrian Rosebrock
November 14, 2019 at 9:13 am
(https://www.pyimagesearch.com/2019/07/15/video-classification-with-keras-and-deep-learning/#comment-573209)

You're using TensorFlow 2.0. In TF 2.0 they changed the the key names. I'll be updating this post soon to cover TF 2.0.

Md. Imrul Hasan
November 15, 2019 at 11:09 am
(https://www.pyimagesearch.com/2019/07/15/video-classification-with-keras-and-deep-learning/#comment-573932)

Thanks Sir..... Everything ran well. which portion of the code should be changed if I could use CNN+LSTM to do the Video Classification?

Adrian Rosebrock
November 21, 2019 at 9:22 am
(https://www.pyimagesearch.com/2019/07/15/video-classification-with-keras-and-deep-learning/#comment-576858)

I'll be doing a separate tutorial on RNNs and LSTMs in the future.

Vitaliy
November 15, 2019 at 4:21 pm
(https://www.pyimagesearch.com/2019/07/15/video-classification-with-keras-and-deep-learning/#comment-574044)

Hi Adrian,
I tried to accomplish a fast run of the algorithm but there is no model with the code. Do you have full code or something that works? Thank you

---

Adrian Rosebrock
November 21, 2019 at 9:21 am
(https://www.pyimagesearch.com/2019/07/15/video-classification-with-keras-and-deep-learning/#comment-576853)

You can train and generate the model using the source code provided with this tutorial.

Jason
November 15, 2019 at 11:54 pm
(https://www.pyimagesearch.com/2019/07/15/video-classification-with-keras-and-deep-learning/#comment-574162)

Hi Adrian, thanks for the tutorial.

I have a question about classifying multiple sports in a single video. Because the sample video you presented only includes one type of sport, I wonder have you tried to train the model on videos with two or more categories of sports? How is the performance?

Also, I wonder if RNN and LTSMs are suitable for real-time image classification?

Thank you!

Adrian Rosebrock
November 21, 2019 at 9:21 am (https://www.pyimagesearch.com/2019/07/15/video-classification-with-keras-and-deep-learning/#comment-576850)

Yes, RNNs and LSTMs can be used. I'll be doing a separate tutorial on them in the future.

Phoenix
November 24, 2019 at 1:12 am (https://www.pyimagesearch.com/2019/07/15/video-classification-with-keras-and-deep-learning/#comment-581063)

This model trained on sports data-set. How can I able to build a general video classification model based on any activity? Any tips??

Thanks in advance.

Adrian Rosebrock
December 5, 2019 at 10:59 am (https://www.pyimagesearch.com/2019/07/15/video-classification-with-keras-and-deep-learning/#comment-589912)

> What you're referring to is called "activity recognition". I'll be demonstrating how to train your own custom activity recognition model in a future tutorial.

Anja
December 17, 2019 at 5:55 am (https://www.pyimagesearch.com/2019/07/15/video-classification-with-keras-and-deep-learning/#comment-596577)

Hi Adrian,

I have a question about using the model.

I created a model using the train method.
Video fildes are labeled correctly.

Can I also use the model for labeling in live cam Steam?
Or does the model have to be trained differently for this?

Anja
December 17, 2019 at 12:06 pm
(https://www.pyimagesearch.com/2019/07/15/video-classification-with-keras-and-deep-learning/#comment-596934)

Hi Adrian,

I have a question about the code:

I have modified the code from you so that I can now access a WebCam (Notbook)
to label live pictures. Everything is going well ... but

The function that reads the label from my model delays the display in the webcam:

Code position:
#make predictions on the frame and then update the predictions
#queue
preds = model.predict (np.expand_dims (frame, axis = 0)) [0]
Q.append (preds)

When I comment out the code, the image and movements in front of the camera are shown in real time.
So the delayed display is due to these lines of code.

Do you have any ideas how I can change this?

Thanks in advance 🙂

Adrian Rosebrock
December 18, 2019 at 9:46 am (https://www.pyimagesearch.com/2019/07/15/video-classification-with-keras-and-deep-learning/#comment-598370)

Calling “model.predict” is NOT a free operation. It’s running neural network that is computationally expensive. That code needs to be included in order to make predictions.

If you would like to speedup the “model.predict” call, consider using a GPU.

Anja
December 18, 2019 at 2:20 pm (https://www.pyimagesearch.com/2019/07/15/video-classification-with-keras-and-deep-learning/#comment-598969)

hi Adrian,

many thanks for your response. Since I'm a beginner in the field, I don't know exactly what to do with:

"Included in order model.predict predictions"
Mean?
Do you have an example of how to do this?

I would like to use the trained model for live monitoring.
If the model works well, I will install the code on a Raspberry PI. At the moment everything is still running on a Windows system.

Adrian Rosebrock
December 26, 2019 at 10:09 am (https://www.pyimagesearch.com/2019/07/15/video-classification-with-keras-and-deep-learning/#comment-617935)

> It's okay if you are new to the field but I would suggest you walk before you run. I recommend you read **Deep Learning for Computer Vision with Python (https://www.pyimagesearch.com/deep-learning-computer-vision-python-book/)**

first — that book will teach you how to apply deep learning to your own images and video. Once you have a strong foundation you can continue with your project.

MVK
January 21, 2020 at 3:34 pm (https://www.pyimagesearch.com/2019/07/15/video-classification-with-keras-and-deep-learning/#comment-667620)

Hello Adrian,

Thank you for this wonderful tutorial.
I tried your tutorial on my laptop, it took too much time, so I tried your code on GPU(GTX1050Ti) it is showing me “ResourceExhaustedError”.
Is there any other way or modification to run this code on GPUs. (steps/software I should follow)
Thank you

Adrian Rosebrock
January 23, 2020 at 9:20 am (https://www.pyimagesearch.com/2019/07/15/video-classification-with-keras-and-deep-learning/#comment-671588)

> When did you receive that error? During training? Or during inference/prediction?

Mounir
January 22, 2020 at 8:39 am (https://www.pyimagesearch.com/2019/07/15/video-classification-with-keras-and-deep-learning/#comment-669254)

Hi Adrian,

Thank you for this article. I saw that you applied the Keras ImageDataGenerator to perform data augmentation on image data.

My question is the following :
Could you, please, tell me if you apply this on a sequence of images picked from a video, will it apply the same type of data augmentation on all the images of this same video ?

Nandakishor
January 25, 2020 at 12:07 pm (https://www.pyimagesearch.com/2019/07/15/video-classification-with-keras-and-deep-learning/#comment-673881)

Hi Adrian ,
Thank you for your tutorial, I found put that there is no folder named “data” in Sports-Type-Classifier. can you give me some suggestions?

Sovit Ranjan Rath (https://debuggercafe.com/)
January 25, 2020 at 12:34 pm
(https://www.pyimagesearch.com/2019/07/15/video-classification-with-keras-and-deep-learning/#comment-673947)

Hi Adrian, amazing post.

By the way, is it okay if I implement the content of this article using PyTorch? I mean I will be using the same dataset for training as yours. I will be posting the PyTorch video classification article on my blog/website, **https://debuggercafe.com/ (https://debuggercafe.com/)**.

Regards,
Sovit

Adrian Rosebrock
January 30, 2020 at 8:53 am
(https://www.pyimagesearch.com/2019/07/15/video-classification-with-keras-and-deep-learning/#comment-689029)

Yes, provided that you link back to the original blog post here on PyImageSearch and **properly cite it. (https://www.pyimagesearch.com/faqs/single-faq/how-do-i-reference-or-cite-one-of-your-blog-posts-books-or-courses)**

Sovit Ranjan Rath (https://debuggercafe.com/.)
February 3, 2020 at 9:40 pm
(https://www.pyimagesearch.com/2019/07/15/video-classification-with-keras-and-deep-learning/#comment-697959)

Sure, I will cite it. And thanks for allowing me to do it.

Kishor
January 25, 2020 at 1:06 pm
(https://www.pyimagesearch.com/2019/07/15/video-classification-with-keras-and-deep-learning/#comment-674025)

Dear Adrian,
thank you for the wonderful tutorial. I implemented this on Colab since my local system not able to handle the training and its great

Lakjeewa Wijebandara
February 6, 2020 at 10:53 am
(https://www.pyimagesearch.com/2019/07/15/video-classification-with-keras-and-deep-learning/#comment-707020)

So considering a cricket shot, how can I model up to focus on the technique of the cricket shot? Say,
player is striking a cricket shot and i want to detect the false he made and what went good in the strike. Can you give me a bit of advice regarding that?

Osman
February 12, 2020 at 1:56 am
(https://www.pyimagesearch.com/2019/07/15/video-classification-with-keras-and-deep-learning/#comment-722098)

hey Adrian,
why is the first part of the tutorial removed?
I just can see the results only

Adrian Rosebrock
February 13, 2020 at 11:02 am
(https://www.pyimagesearch.com/2019/07/15/video-classification-with-keras-and-deep-learning/#comment-725634)

Thanks for pointing this out, Osman. When we migrated to the new site design I think there may have been an issue. I will investigate.

EDIT: The issue has now been resolved.

Suyash
February 13, 2020 at 10:29 am (https://www.pyimagesearch.com/2019/07/15/video-classification-with-keras-and-deep-learning/#comment-725532)

Can you make an article expaining training part of the code

Adrian Rosebrock
February 13, 2020 at 10:56 am (https://www.pyimagesearch.com/2019/07/15/video-classification-with-keras-and-deep-learning/#comment-725613)

I would suggest you read **Deep Learning for Computer Vision with Python (https://www.pyimagesearch.com/deep-learning-computer-vision-python-book/)** if you would like to learn how to train the model.

Jafar Sadik
February 21, 2020 at 12:27 pm
(https://www.pyimagesearch.com/2019/07/15/video-classification-with-keras-and-deep-learning/#comment-750514)

Hey Adrian,
Really a good tutorial. But the code you sent me over email, has empty model folder. As you said there should have activity.model & lb.pickle files, I found nothing there. Everything else in the directory was fine. Can you please check this issue?

Jafar Sadik
February 21, 2020 at 3:17 pm
(https://www.pyimagesearch.com/2019/07/15/video-classification-with-keras-and-deep-learning/#comment-751037)

Sorted it out. Thanks…

Adrian Rosebrock
February 27, 2020 at 9:25 am
(https://www.pyimagesearch.com/2019/07/15/video-classification-with-keras-and-deep-learning/#comment-759324)

Congrats on resolving the issue, Jafar!

Marco Tolino
February 25, 2020 at 12:18 pm
(https://www.pyimagesearch.com/2019/07/15/video-classification-with-keras-and-deep-learning/#comment-759111)

Hey Adrian,

Very interesting post!

I have a simple question: if I run the training for sport X and Y, creating the relative models and at a later time I want to add sport Z, shall I re-run the training for the whole sports (X,Y,Z) or I can just run the training for the new sport (Z) and select the same output files of the previous training (X,Y)?

Thank you!

Adrian Rosebrock
February 27, 2020 at 9:14 am
(https://www.pyimagesearch.com/2019/07/15/video-classification-with-keras-and-deep-learning/#comment-759308)

You have two options:

1. Look into incremental learning methods
2. Fine-tune the model on the original X and Y classes plus your new Z class

n fs
March 1, 2020 at 6:52 pm
(https://www.pyimagesearch.com/2019/07/15/video-classification-with-keras-and-deep-learning/#comment-764493)

When I git cloned the Sports-Type-Classifier folder there is NO “data” directory in there. So when I type in the tree command it doesn’t work, and I’m stuck there and cannot move further. Where is the “data” directory?

Md. Imrul Hasan
March 9, 2020 at 10:40 pm
(https://www.pyimagesearch.com/2019/07/15/video-classification-with-keras-and-deep-learning/#comment-765892)

Do you publish any paper on it? if so then please give me the link.

Adrian Rosebrock
March 11, 2020 at 4:50 pm
(https://www.pyimagesearch.com/2019/07/15/video-classification-with-keras-and-deep-learning/#comment-766048)

This tutorial serves as the documentation. If you want to reference it, **you can find instructions on how to do so here. (https://www.pyimagesearch.com/faqs/single-faq/how-do-i-reference-or-cite-one-of-your-blog-posts-books-or-courses)**

Md. Imrul Hasan
March 12, 2020 at 5:42 am (https://www.pyimagesearch.com/2019/07/15/video-classification-with-keras-and-deep-learning/#comment-766110)

thanks … I will refer to you. ..

Nick James
March 21, 2020 at 3:14 pm (https://www.pyimagesearch.com/2019/07/15/video-classification-with-keras-and-deep-learning/#comment-767156)

Where do you get the mean values from in line 101?

mean = np.array([123.68, 116.779, 103.939], dtype=”float32″)

Adrian Rosebrock
March 25, 2020 at 1:43 pm (https://www.pyimagesearch.com/2019/07/15/video-classification-with-keras-and-deep-learning/#comment-767658)

> They are used for mean subtraction and scaling. Those values are the RGB means computed across the ImageNet dataset (which is what the model was originally trained on).

Dang Tuan Hoang
March 25, 2020 at 4:30 am (https://www.pyimagesearch.com/2019/07/15/video-classification-with-keras-and-deep-learning/#comment-767576)

Hi Adrian,
I'm working on an AI to perform sensitive-content (violence, pornography, ...) detection in videos. How should I interpret the output in multiple categories presented in the video? ATM, I'm detecting all the frames, and save max confidence value for each category, and return them as the probability of that category appear in the video.

Before you leave a comment...

Hey, Adrian here, author of the PyImageSearch blog. I'd love to hear from you; however, I have made the decision to no longer offer free 1:1 help over blog post comments. I simply do not have the time to

moderate and respond to them all.

To that end, myself and my team are doubling down our efforts on supporting our paying customers, writing new books and courses, and authoring high quality Computer Vision, Deep Learning, and OpenCV content for you to learn from.

I'd be happy to help you with your question or project, but **I have to politely ask you to purchase one of my books or courses first. (https://www.pyimagesearch.com/books-and-courses/)**

Why bother becoming a PyImageSearch customer?

- You'll receive a **great education** through my premium content
- You'll receive **priority support**
- You'll receive a **guaranteed response** from myself and my team
- You'll be able to **confidently and successfully** apply Computer Vision, Deep Learning, and OpenCV to your projects

Click here to see my full catalog of books and courses. (https://www.pyimagesearch.com/books-and-courses/) Take a look and I hope to see you on the other side!

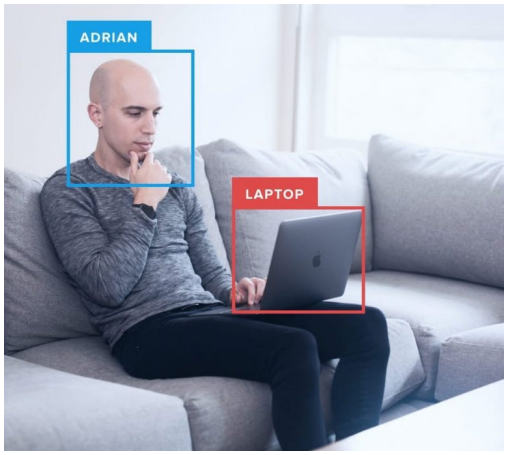
Similar articles

ANOMALY/OUTLIER DETECTION DEEP LEARNING

KERAS AND TENSORFLOW TUTORIALS

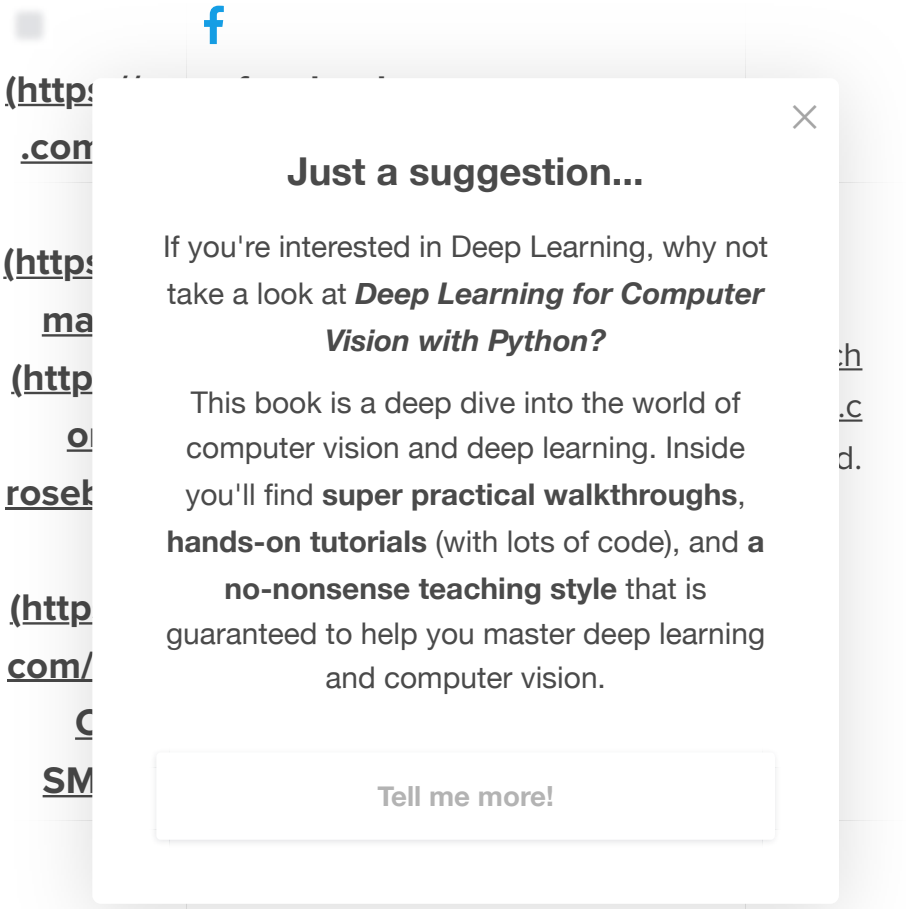
Anomaly detection with Keras, TensorFlow, and Deep Learning (https://www.pyimagesearch.com/2020/03/02/anomaly-detection-with-keras-tensorflow-and-deep-learning/)

March 2, 2020



You can learn Computer Vision, Deep Learning, and OpenCV.

Get your FREE 17 page Computer Vision, OpenCV, and Deep Learning Resource Guide PDF. Inside you'll find my hand-picked tutorials, books, courses, and libraries to help you master CV and DL.


Just a suggestion...
If you're interested in Deep Learning, why not take a look at *Deep Learning for Computer Vision with Python?*
This book is a deep dive into the world of computer vision and deep learning. Inside you'll find **super practical walkthroughs**, **hands-on tutorials** (with lots of code), and **a no-nonsense teaching style** that is guaranteed to help you master deep learning and computer vision.
Tell me more!