

# Sports Image Classification

## Deep Learning

## Data Preparation:

### Classification Model convnet\_cifar10 and VGG:

Using **Image Preprocessing** class

- Normalize a picture pixel to 0-1 float (instead of 0-255 int).
- Add sample wise zero center (Zero center each sample by subtracting it by its mean).
- Add feature wise stdnorm (Scale each sample by the specified standard deviation. If no specified, std is evaluated over all samples data).

```
img_prep = ImagePreprocessing()  
img_prep.add_featurewise_zero_center()  
img_prep.add_featurewise_stdnorm()
```

*Figure 1:Image Preprocessing*

Using **Image Augmentation** class

This class is meant to be used as an argument of `input\_data`. When training a model, the defined augmentation methods will be applied at training time only. Note that Image Preprocessing is like Image Augmentation but applies at both training time and testing time.

- Add random flip left right
- Add random rotation Randomly rotate an image by a random angle (-max\_angle, max\_angle).

```
img_aug = ImageAugmentation()  
img_aug.add_random_flip_leftright()  
img_aug.add_random_rotation(max_angle=25.)
```

*Figure 2:Image Augmentation*

## Models' descriptions and techniques:

### Classification using Convolutional neural network model Convnet architecture:

- Use **'tflearn'** modules to train the model to classify between different persons' signatures
- Use **One-hot encoding** technique to label our training data
  - Basketball -> [1,0,0,0,0,0]
  - Football -> [0,1,0,0,0,0]
  - Rowing -> [0,0,1,0,0,0]
  - Swimming -> [0,0,0,1,0,0]
  - Tennis -> [0,0,0,0,1,0]
  - Yoga -> [0,0,0,0,0,1]
- Use convnet\_cifar10 CNN architecture
- Use **softmax** as activation function in output layer with 5 neurons.

```
def create_label(image_name):  
    """ ... """  
    word_label = image_name.split('.')[0]  
    # if "Basketball" in word_label  
    if word_label.__contains__('Basketball'):  
        return np.array([1, 0, 0, 0, 0, 0])  
    elif word_label.__contains__('Football'):  
        return np.array([0, 1, 0, 0, 0, 0])  
    elif word_label.__contains__('Rowing'):  
        return np.array([0, 0, 1, 0, 0, 0])  
    elif word_label.__contains__('Swimming'):  
        return np.array([0, 0, 0, 1, 0, 0])  
    elif word_label.__contains__('Tennis'):  
        return np.array([0, 0, 0, 0, 1, 0])  
    elif word_label.__contains__('Yoga'):  
        return np.array([0, 0, 0, 0, 0, 1])
```

Figure 3:One hot encoding

```
network = input_data(shape=[None, IMG_SIZE, IMG_SIZE, 3],  
                     data_preprocessing=img_prep,  
                     data_augmentation=img_aug)  
network = conv_2d(network, 32, 3, activation='relu')  
network = max_pool_2d(network, 2)  
network = conv_2d(network, 64, 3, activation='relu')  
network = conv_2d(network, 64, 3, activation='relu')  
network = max_pool_2d(network, 2)  
network = fully_connected(network, 512, activation='relu')  
network = dropout(network, 0.5)  
network = fully_connected(network, 6, activation='softmax')  
network = regression(network, optimizer='adam',  
                     loss='categorical_crossentropy',  
                     learning_rate=0.00005)  
  
model = tflearn.DNN(network, tensorboard_verbose=0)  
model.fit(X_train, y_train, n_epoch=150, show_metric=True, validation_set=0.2)
```

Figure 4: Convnet Model Architecture

## ConvNet Tensorboard:

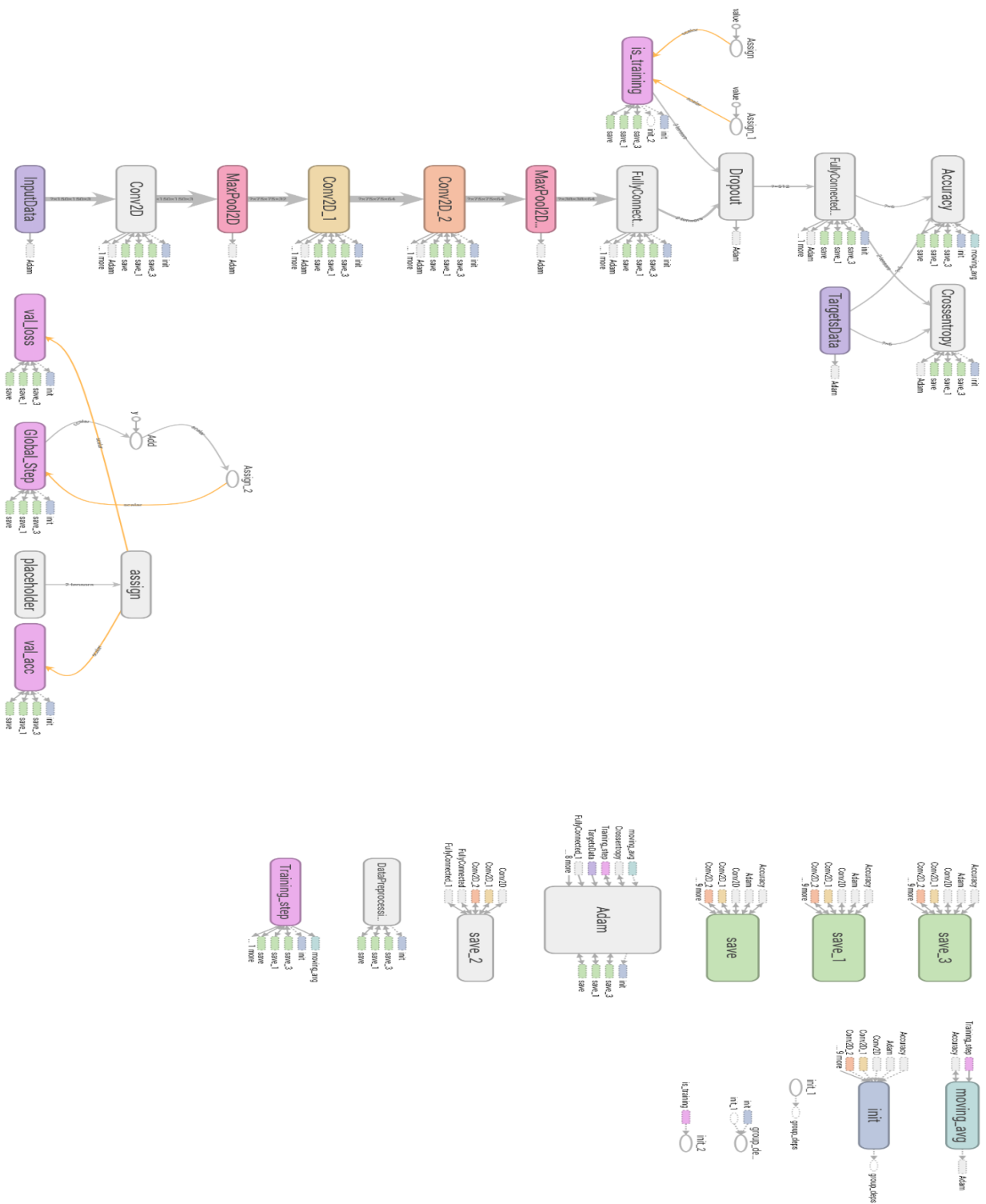


Figure 5: Model Tensorboard

## Classification using VGG model:

- Use 'tflearn' modules to train the model to classify between different persons' signatures
- Use **One-hot encoding** technique to label our training data
  - Basketball -> [1,0,0,0,0]
  - Football -> [0,1,0,0,0]
  - Rowing -> [0,0,1,0,0]
  - Swimming -> [0,0,0,1,0]
  - Tennis -> [0,0,0,0,1]
  - Yoga -> [0,0,0,0,0]
- Use VGG CNN architecture
- Use **softmax** as activation function in output layer with 5 neurons.

```
123 def vggModel():
124     network = input_data(shape=[None, IMG_SIZE, IMG_SIZE, 3], data_preprocessing=img_prep, data_augmentation=img_aug)
125     network = conv_2d(network, 64, 3, activation='relu')
126     network = conv_2d(network, 64, 3, activation='relu')
127     network = max_pool_2d(network, 2)
128     network = conv_2d(network, 128, 3, activation='relu')
129     network = conv_2d(network, 128, 3, activation='relu')
130     network = max_pool_2d(network, 2)
131     network = conv_2d(network, 256, 3, activation='relu')
132     network = conv_2d(network, 256, 3, activation='relu')
133     network = conv_2d(network, 256, 3, activation='relu')
134     network = conv_2d(network, 256, 3, activation='relu')
135     network = max_pool_2d(network, 2)
136     network = conv_2d(network, 512, 3, activation='relu')
137     network = conv_2d(network, 512, 3, activation='relu')
138     network = conv_2d(network, 512, 3, activation='relu')
139     network = conv_2d(network, 512, 3, activation='relu')
140     network = max_pool_2d(network, 2)
141     network = conv_2d(network, 512, 3, activation='relu')
142     network = conv_2d(network, 512, 3, activation='relu')
143     network = conv_2d(network, 512, 3, activation='relu')
144     network = conv_2d(network, 512, 3, activation='relu')
145     network = max_pool_2d(network, 2)
146     network = conv_2d(network, 1024, 3, activation='relu')
147     network = conv_2d(network, 1024, 3, activation='relu')
148     network = max_pool_2d(network, 2)
149     network = fully_connected(network, 2024, activation='relu')
150     network = fully_connected(network, 2024, activation='relu')
151     network = dropout(network, 0.5)
152     network = fully_connected(network, 6, activation='softmax')
153     network = regression(network, optimizer='adam', loss='categorical_crossentropy', learning_rate=LR)
154     model = tflearn.DNN(network, tensorboard_verbose=0)
155     return model
```

Figure 6: VGG Model

## ConvNet Model Accuracy:

### Train Accuracy:

- Train data -> 1344
- Validation data -> 337

```
Training Step: 3148 | total loss: 0.33535 | time: 9.344s
| Adam | epoch: 150 | loss: 0.33535 - acc: 0.9453 -- iter: 1216/1344
Training Step: 3149 | total loss: 0.32584 | time: 9.848s
| Adam | epoch: 150 | loss: 0.32584 - acc: 0.9430 -- iter: 1280/1344
Training Step: 3150 | total loss: 0.30425 | time: 11.359s
| Adam | epoch: 150 | loss: 0.30425 - acc: 0.9456 | val_loss: 0.52134 - val_acc: 0.8368 -- iter: 1344/1344
```

Figure 7: ConvNet Train and validation accuracy

Train accuracy = 94.56%

Validation accuracy = 83.68%

### Test Accuracy:

- Test data -> 688



Sports\_kaggle\_M1\_150epoc\_00005lr\_aug.csv  
Complete · Youssef Nader Michel · 10d ago

0.80072

0.83009



Figure 8: ConvNet test accuracy

Test accuracy = 83.009%

## VGG Model accuracy:

### Train accuracy:

- Train data -> 1344
- Validation data -> 337



The screenshot shows a Kaggle notebook titled "Sports\_CNN". The code in the notebook is as follows:

```
OutPut_list = []
for i in range(len(pred)):
    x = [images_names[i], pred[i]]
    OutPut_list.append(x)
with open("Sports.csv", "w") as Sport:
    student = csv.writer(Sport)
    student.writerow(headers)
    student.writerows (OutPut_list)

import pandas as pd
df = pd.read_csv('Sports.csv')
print(df)
print("Vgg Second MODEL \n\n")
```

The output of the notebook shows the training progress and the final accuracy:

```
Training Step: 394 | total loss: 0.82780 | time: 1.580s
| Adam | epoch: 029 | loss: 0.82780 - acc: 0.7099 -- iter: 0256/1681
```

The output also shows the final accuracy of the second model:

```
0.7099
```

Train accuracy = 70.99%

### Test accuracy:

- Test data -> 688



**Sports.csv**

Complete · Ahmed Gamal ibrahim ali · 11d ago

**0.83333**

**0.81553**



Test accuracy = 83.3%

## Trials and other models:

Model Name	Model details	Train accuracy	Validation accuracy	Test accuracy
ConvNet	<ul style="list-style-type: none"> <li>Epochs = 150</li> <li>LR = 0.001</li> <li>IMG_S = 50</li> </ul>	0.9935	0.8056	0.76449
network_in_network	<ul style="list-style-type: none"> <li>Epochs = 100</li> <li>LR = 0.001</li> <li>IMG_S = 50</li> </ul>	0.7548	0.5058	0.69174
Another archi for ConvNet	<ul style="list-style-type: none"> <li>Epochs = 150</li> <li>LR = 0.001</li> <li>IMG_S = 50</li> </ul>	0.9277	0.7596	0.50242
Inception	<ul style="list-style-type: none"> <li>Epoch = 1000</li> <li>LR = 0.0001</li> <li>IMG_S = 50</li> </ul>	0.9565	0.9156	0.80582

## Another trials and submissions:

■ <https://www.kaggle.com/competitions/n23-sports-image-classification/submissions#>

**Version 18**  
 Quick Version • Diff: +4 -4  
 Ran in 3 seconds

9d ago

**Version 17**  
 Quick Version • Diff: +2 -2  
 Ran in 2 seconds

11d ago

**Version 16**  
 Quick Version • Diff: +6 -8  
 Ran in 3 seconds

11d ago

**Version 15**  
 Quick Version • Diff: +4 -6  
 Ran in 4 seconds

11d ago

**Version 14**  
 Quick Version • Diff: +15 -1  
 Ran in 3 seconds

12d ago

**Version 13**  
 Quick Version • Diff: +0 -0  
 Ran in 3 seconds

13d ago

**Version 12**  
 Save & Run All • Diff: +1 -1  
 Cancelled after 0 seconds

13d ago

**version\_11\_Y**  
 Quick Version • Diff: +28 -5  
 Ran in 2 seconds

13d ago

**version\_10\_Y**  
 Quick Version • Diff: +24 -8  
 Ran in 3 seconds

13d ago

**Version 9**  
 Save & Run All • Diff: +0 -0  
 Failed after 25 seconds

14d ago

**Version 8\_G**  
 Save & Run All • Diff: +2 -0  
 Failed after 28 seconds

14d ago

**Version 7**  
 Quick Version • Diff: +1 -5  
 Ran in 3 seconds

14d ago

**Version 6**  
 Save & Run All • Diff: +0 -0  
 Failed after 33 seconds

14d ago

**version\_5\_Y**  
 Save & Run All • Diff: +11 -12  
 Failed after 33 seconds

14d ago

**version\_4\_Y**

14d ago

**Version History**

Quick Version • Diff: +1 -1  
 Ran in 2 seconds

3d ago

**Version 7**  
 Quick Version • Diff: +7 -4  
 Ran in 3 seconds

3d ago

**Version 6**  
 Save & Run All • Diff: +11 -11  
 Failed after 1 hour and 13 minutes

3d ago

**Version 5**  
 Quick Version • Diff: +21 -24  
 Ran in 2 seconds

5d ago

**Version 4**  
 Quick Version • Diff: +31 -15  
 Ran in 3 seconds

5d ago

**Version 3**  
 Quick Version • Diff: +33 -31  
 Ran in 3 seconds

5d ago

**Version 2**  
 Quick Version • Diff: +15 -3  
 Ran in 3 seconds

6d ago

**Version 1**  
 Quick Version • Diff: +150 -0  
 Ran in 3 seconds

6d ago