2023
2022

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 -> [0,0,0,0,01
- 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:

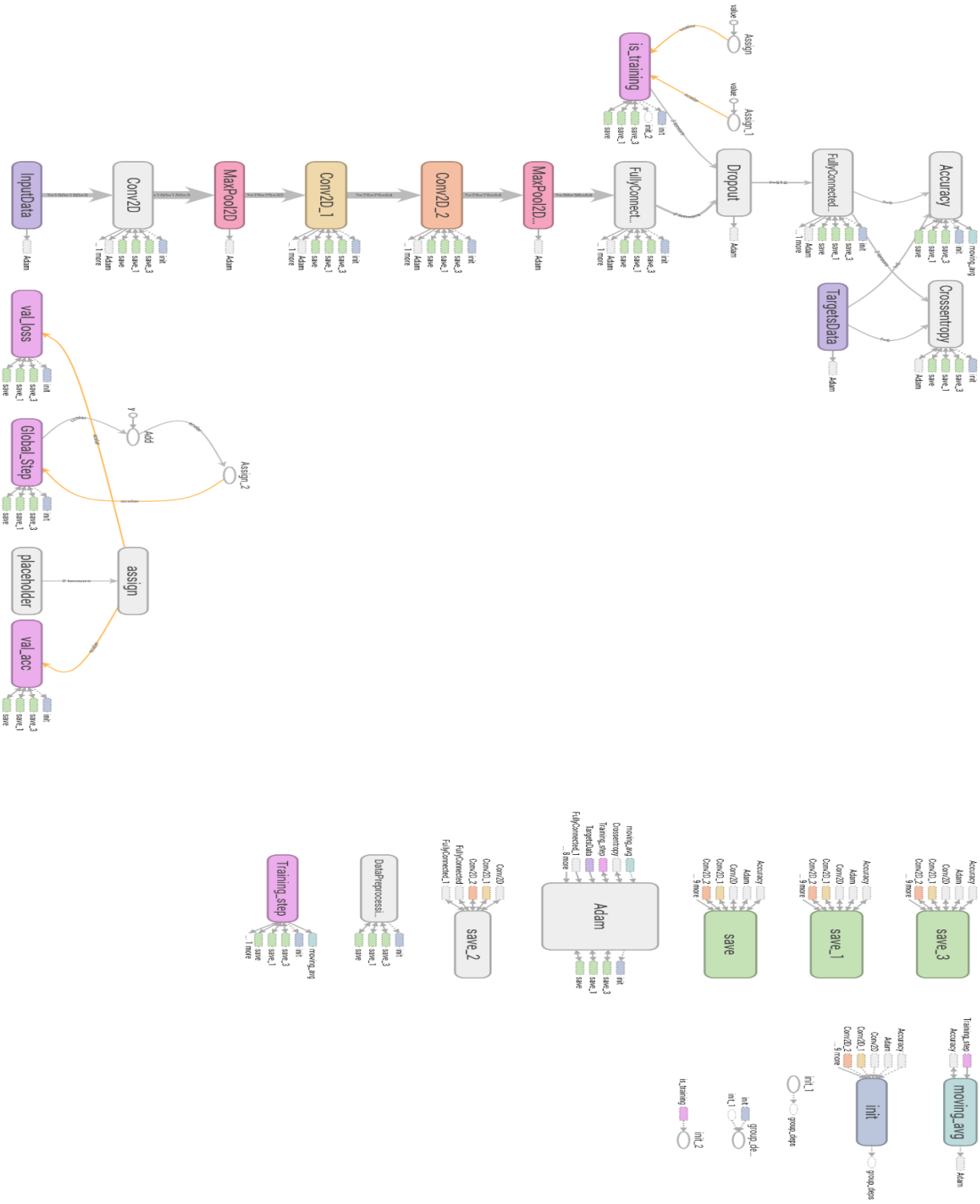


Figure 5: Model Tensorboard

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 -> [1,0,0,0,0,0]
    - 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 -> [0,0,0,0,01
- Use VGG CNN architecture
- Use **softmax** as activation function in output layer with 5 neurons.

```
def vggModel():
    network = input_data(shape=[None, IMG_SIZE, IMG_SIZE, 3], data_preprocessing=img_prep,data_augmentation=img_aug)
    network = conv_2d(network, 64, 3, activation='relu')
    network = conv_2d(network, 64, 3, activation='relu')
    network = max_pool_2d(network, 2)
    network = conv_2d(network, 128, 3, activation='relu')
    network = conv_2d(network, 128, 3, activation='relu')
    network = max_pool_2d(network, 2)
    network = conv_2d(network, 256, 3, activation='relu')
    network = conv_2d(network, 256, 3, activation='relu')
    network = conv_2d(network, 256, 3, activation='relu')
    network = conv_2d(network, 256, 3, activation='relu')
    network = max_pool_2d(network, 2)
    network = conv_2d(network, 512, 3, activation='relu')
    network = conv_2d(network, 512, 3, activation='relu')
    network = conv_2d(network, 512, 3, activation='relu')
    network = conv_2d(network, 512, 3, activation='relu')
    network = max_pool_2d(network, 2)
    network = conv_2d(network, 512, 3, activation='relu')
    network = conv_2d(network, 512, 3, activation='relu')
    network = conv_2d(network, 512, 3, activation='relu')
    network = conv_2d(network, 512, 3, activation='relu')
    network = max_pool_2d(network, 2)
    network = conv_2d(network, 1024, 3, activation='relu')
    network = conv_2d(network, 1024, 3, activation='relu')
    network = max_pool_2d(network, 2)
    network = fully_connected(network, 2024, activation='relu')
    network = fully_connected(network, 2024, activation='relu')
    network = dropout(network, 0.5)
    network = fully_connected(network, 6, activation='softmax')
    network = regression(network, optimizer='adam', loss='categorical_crossentropy',learning_rate=LR)
    model = tflearn.DNN(network, tensorboard_verbose=0)
    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

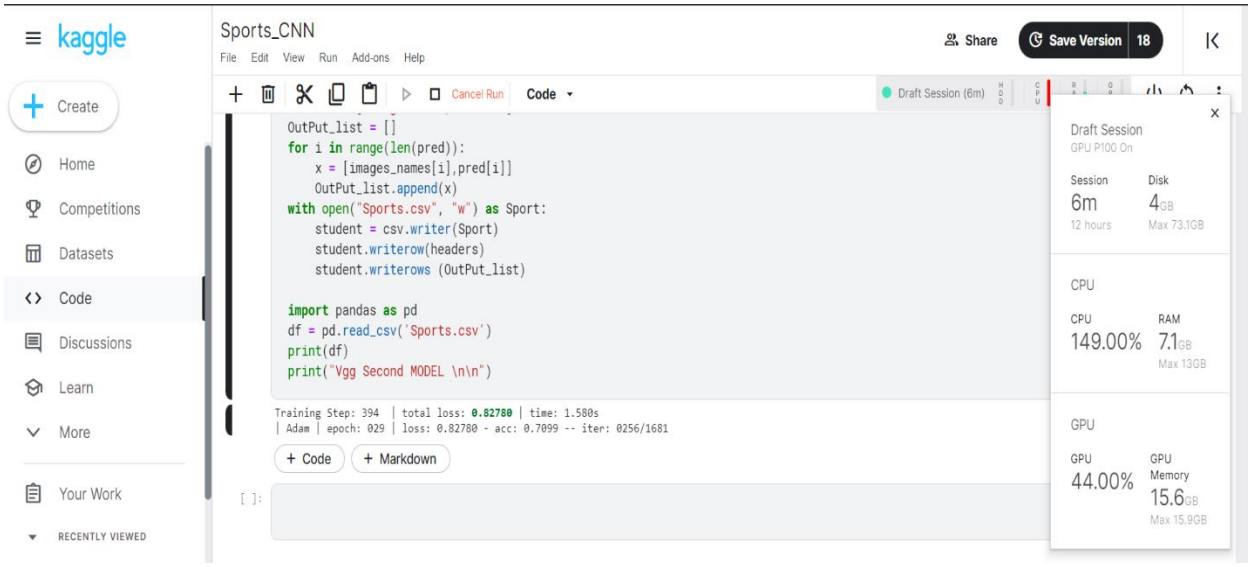
Figure 8:ConvNet test accuracy

Test accuracy = 83.009%

VGG Model accuracy:

Train accuracy:

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



Train accuracy = 70.99%

Test accuracy:

- Test data -> 688

Test accuracy = 83.3%

Trials and other models:

Model Name	Model details	Train accuracy	Validation accuracy	Test accuracy
ConvNet	■ Epochs = 150 ■ LR = 0.001 ■ IMG_S = 50	0.9935	0.8056	0.76449
network_in_network	■ Epochs = 100 ■ LR = 0.001 ■ IMG_S = 50	0.7548	0.5058	0.69174
Another archi for ConvNet	■ Epochs = 150 ■ LR = 0.001 ■ IMG_S = 50	0.9277	0.7596	0.50242
Inception	■ Epoch = 1000 ■ LR = 0.0001 ■ IMG_S = 50	0.9565	0.9156	0.80582

Another trials and submissions:

- https://www.kaggle.com/competitions/nn23-sports-image-classification/submissions#
-