Deep Learning Competition

Autism disorder classification

Team 20

Mohamed Adel	Mohamed Zaki	Mostafa Khaled	Mennatallah Ibrahim	Nour Eldin Nasser
Cs	Cs	Cs	Cs	Cs
2017170347	2017170337	2017170427	2017170446	2017170480

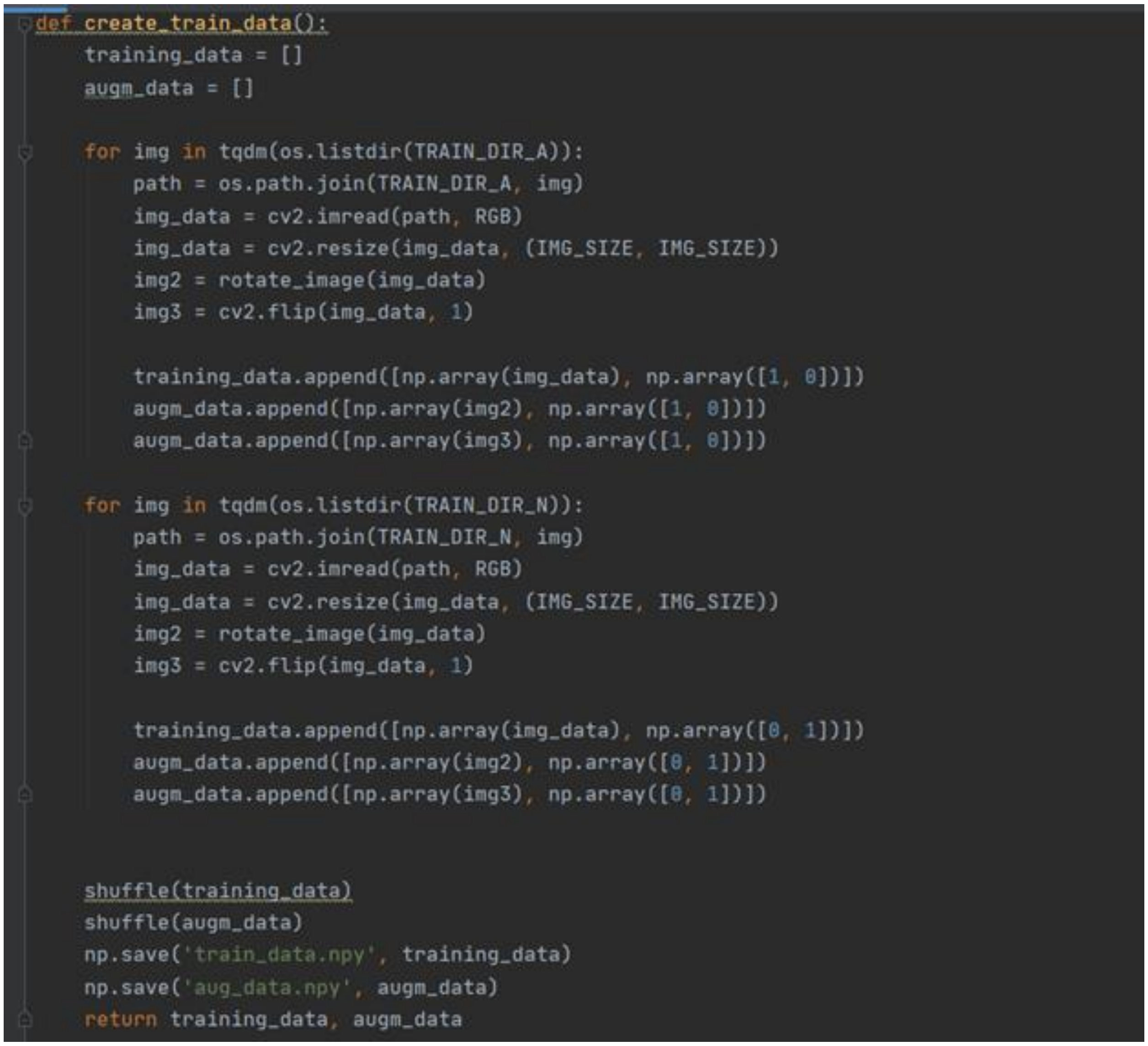
- Introduction :-

  Children with autism have a broader upper face, including wider eyes. Children with autism have a shorter middle region of the face, including the cheeks and nose. Children with autism have a broader or wider mouth and philtrum -- the divot below the nose, above the top lip the, goal is to build deep learning models that can diagnose autism disorder in the children from their face.

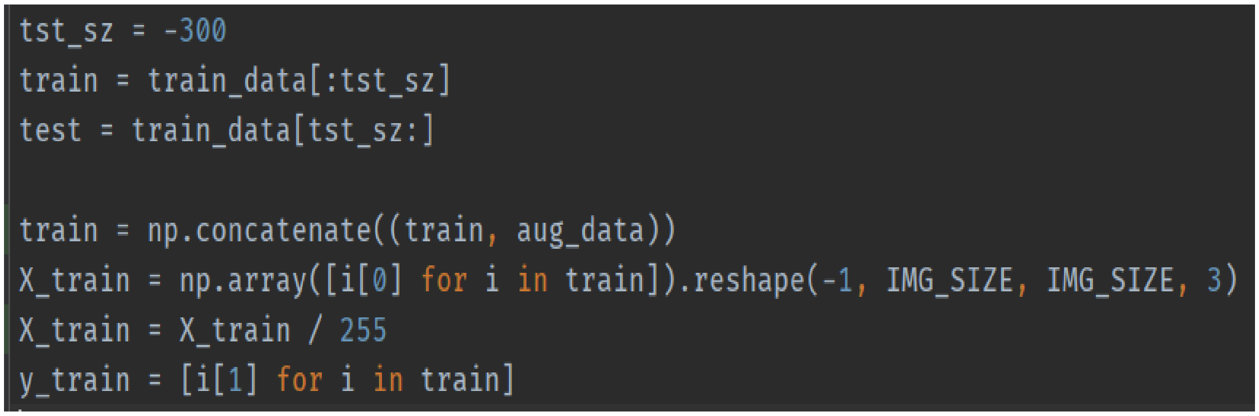
- Dataset :-

  The dataset is organized into 2 folders (train and test) and contains subfolders for the train category (autistic children and non-autistic children). There are 2,536 children faces training images, 1,268 autistic children faces and 1,268 non-autistic children faces. The testing set contains 400 images for children.

- Preprocessing :-
    - Normalized images by dividing each value by 255 to make it from 0 to 1.
    - The images were converted to RGB.
    - Image Data Augmentation : Used Tflearn image augmentation to make the dataset bigger by randomly flipping images left and right and adding random rotation.
    - Dataset splitting : The training data were split into train/validation sets, the validation set has 300 images , the rest was for the train.



```python
def create_train_data():
    training_data = []
    augm_data = []

    for img in tqdm(os.listdir(TRAIN_DIR_A)):
        path = os.path.join(TRAIN_DIR_A, img)
        img_data = cv2.imread(path, RGB)
        img_data = cv2.resize(img_data, (IMG_SIZE, IMG_SIZE))
        img2 = rotate_image(img_data)
        img3 = cv2.flip(img_data, 1)

        training_data.append([np.array(img_data), np.array([1, 0])])
        augm_data.append([np.array(img2), np.array([1, 0])])
        augm_data.append([np.array(img3), np.array([1, 0])])

    for img in tqdm(os.listdir(TRAIN_DIR_N)):
        path = os.path.join(TRAIN_DIR_N, img)
        img_data = cv2.imread(path, RGB)
        img_data = cv2.resize(img_data, (IMG_SIZE, IMG_SIZE))
        img2 = rotate_image(img_data)
        img3 = cv2.flip(img_data, 1)

        training_data.append([np.array(img_data), np.array([0, 1])])
        augm_data.append([np.array(img2), np.array([0, 1])])
        augm_data.append([np.array(img3), np.array([0, 1])])


    shuffle(training_data)
    shuffle(augm_data)
    np.save('train_data.npy', training_data)
    np.save('aug_data.npy', augm_data)
    return training_data, augm_data
```



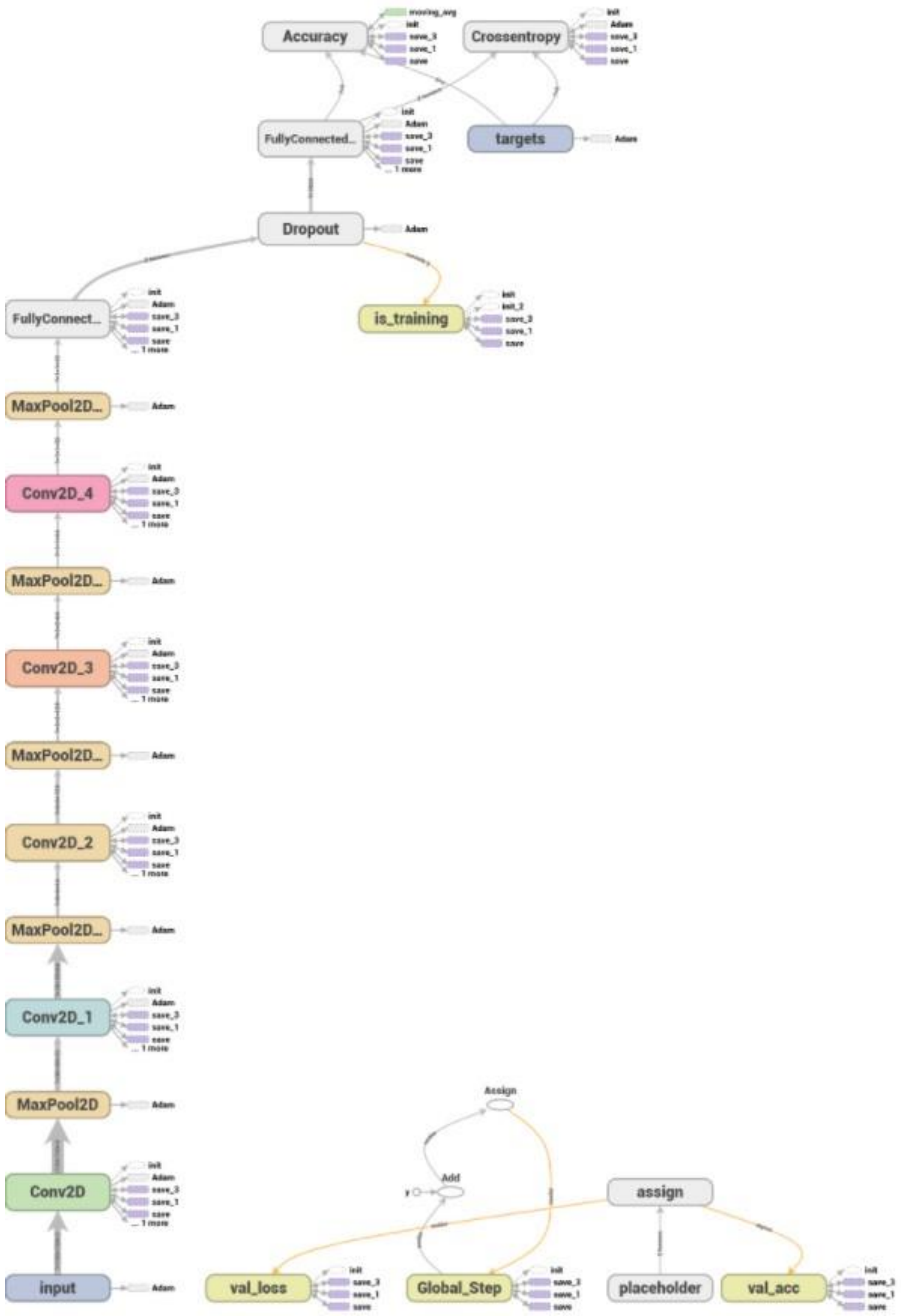
```
tst_sz = -300
train = train_data[:tst_sz]
test = train_data[tst_sz:]

train = np.concatenate((train, aug_data))
X_train = np.array([i[0] for i in train]).reshape(-1, IMG_SIZE, IMG_SIZE, 3)
X_train = X_train / 255
y_train = [i[1] for i in train]
```

- Models :-
  - Convolutional Neural Network (CNN):

    Architecture :-

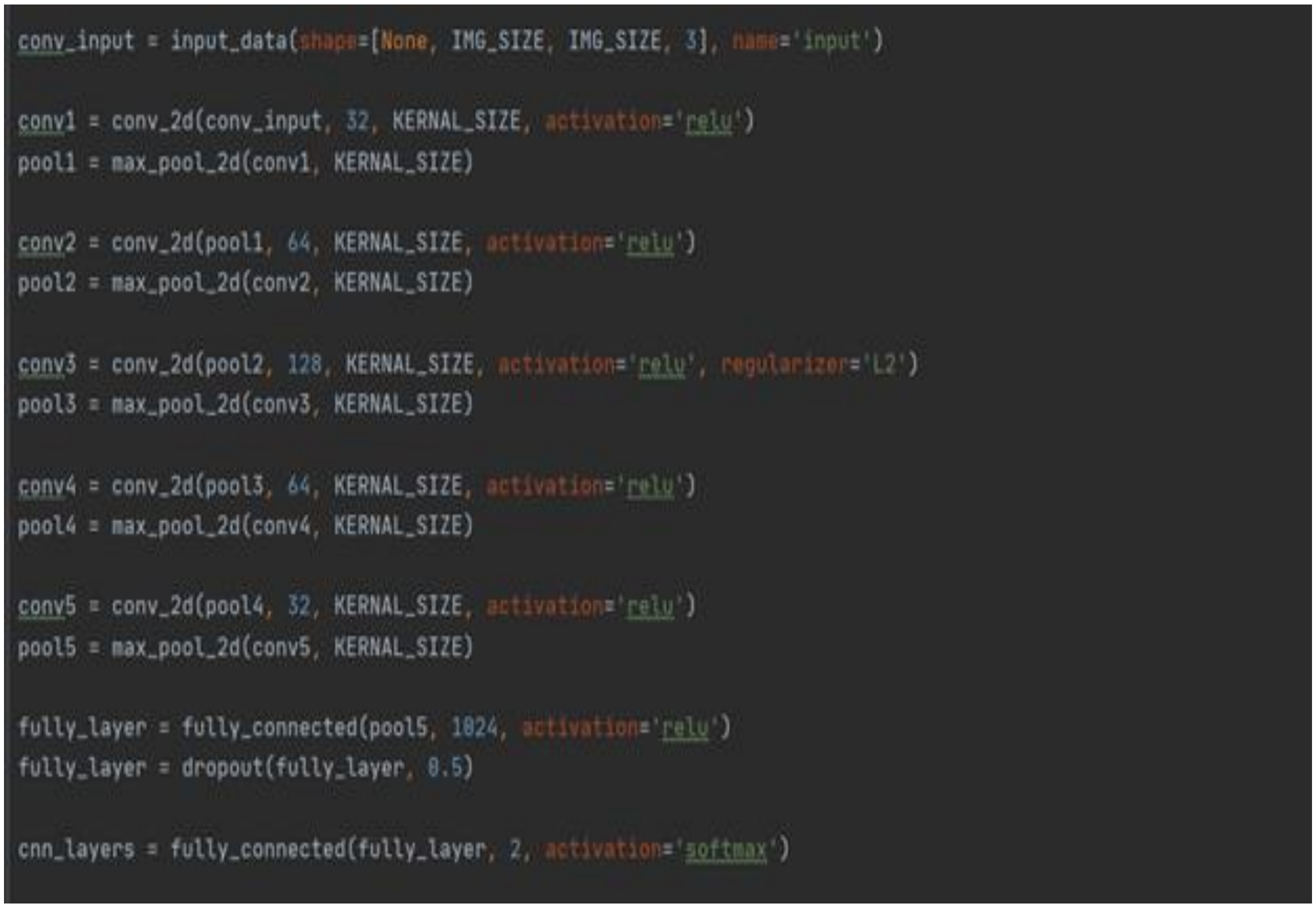
    The model contains 5 layers :

    - First layer is a convolution layer with 32 feature maps and a pooling layer.
    - Second layer is a convolution layer with 64 feature maps and a pooling layer.
    - Third layer is a convolution layer with 128 feature maps and a pooling layer.
    - Fourth layer is a convolution layer with 64 feature maps and a pooling layer .
    - Fifth layer is a convolution layer with 32 feature maps and a pooling layer.
    - Size of filters is 3x3 with stride 1.
    - Fully connected layers then output layer.


Accuracy
moving_avg
init
save_3
save_1
save
Crossentropy
init
Adam
save_3
save_1
save
FullyConnected...
init
Adam
save_3
save_1
save
... 1 more
targets
Adam
Dropout
Adam
FullyConnect...
init
Adam
save_3
save_1
save
... 1 more
is_training
init
init_2
save_3
save_1
save
MaxPool2D...
Adam
Conv2D_4
init
Adam
save_3
save_1
save
... 1 more
MaxPool2D...
Adam
Conv2D_3
init
Adam
save_3
save_1
save
... 1 more
MaxPool2D...
Adam
Conv2D_2
init
Adam
save_3
save_1
save
... 1 more
MaxPool2D...
Adam
Conv2D_1
init
Adam
save_3
save_1
save
... 1 more
MaxPool2D
Adam
Conv2D
init
Adam
save_3
save_1
save
... 1 more
input
Adam
Assign
Add
y
assign
val_loss
init
save_3
save_1
save
Global_Step
init
save_3
save_1
save
placeholder
val_acc
init
save_3
save_1
save

Hyperparameters :-

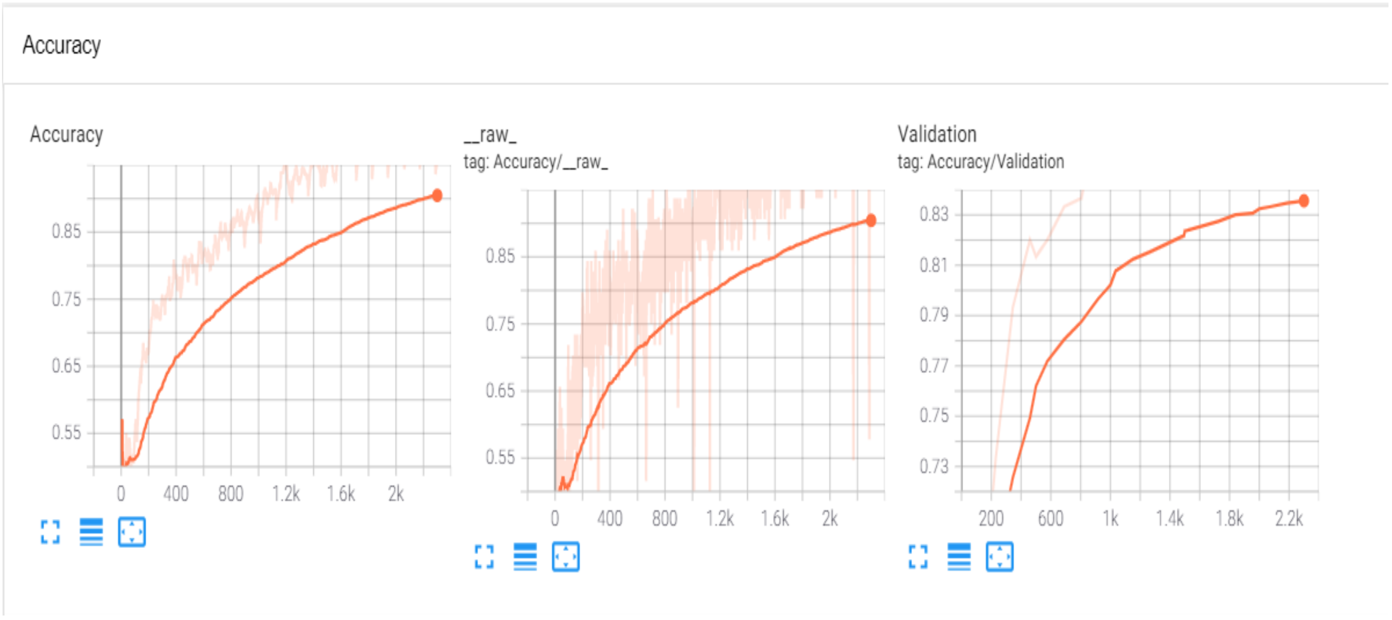
- This model was compiled using Adam optimization algorithm, categorical cross-entropy as its loss function. The dropout layer used has a drop-out probability of 50%. The output layer has softmax as its activation function. The rest of the network uses ReLU.

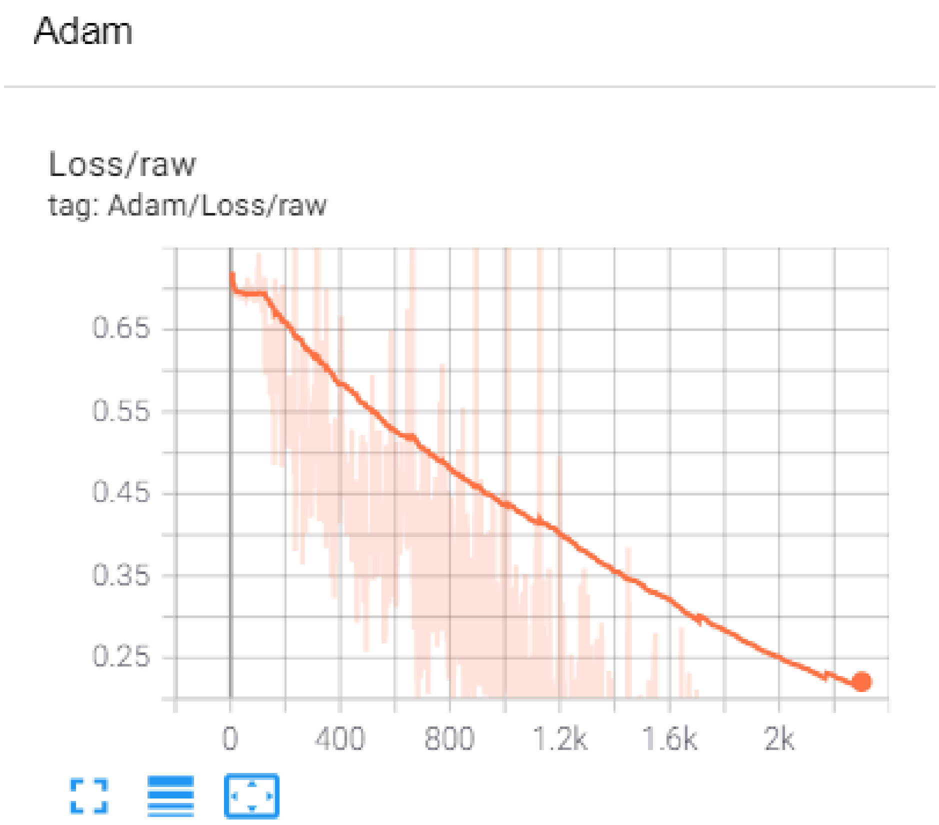


```
conv_input = input_data(shape=[None, IMG_SIZE, IMG_SIZE, 3], name='input')

conv1 = conv_2d(conv_input, 32, KERNAL_SIZE, activation='relu')
pool1 = max_pool_2d(conv1, KERNAL_SIZE)

conv2 = conv_2d(pool1, 64, KERNAL_SIZE, activation='relu')
pool2 = max_pool_2d(conv2, KERNAL_SIZE)

conv3 = conv_2d(pool2, 128, KERNAL_SIZE, activation='relu', regularizer='L2')
pool3 = max_pool_2d(conv3, KERNAL_SIZE)

conv4 = conv_2d(pool3, 64, KERNAL_SIZE, activation='relu')
pool4 = max_pool_2d(conv4, KERNAL_SIZE)

conv5 = conv_2d(pool4, 32, KERNAL_SIZE, activation='relu')
pool5 = max_pool_2d(conv5, KERNAL_SIZE)

fully_layer = fully_connected(pool5, 1024, activation='relu')
fully_layer = dropout(fully_layer, 0.5)

cnn_layers = fully_connected(fully_layer, 2, activation='softmax')
```

Results :-

	Value	Train accuracy	Validation accuracy	Test accuracy
Image Size	150	98%	86%	84%
Epochs	15			
Learning Rate	0.001			
Color	RGB			
Kernel Size	3			

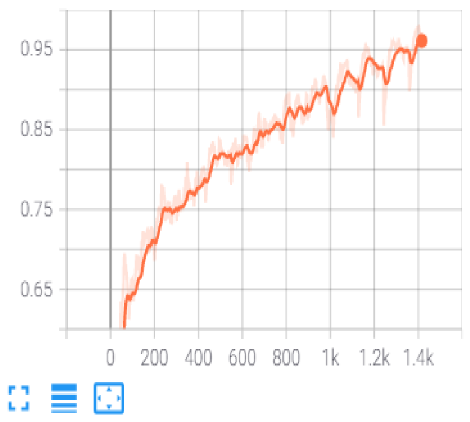

Accuracy
Accuracy
0.85
0.75
0.65
0.55
0
400
800
1.2k
1.6k
2k
raw
tag: Accuracy/_raw_
0.85
0.75
0.65
0.55
0
400
800
1.2k
1.6k
2k
Validation
tag: Accuracy/Validation
0.83
0.81
0.79
0.77
0.75
0.73
200
600
1k
1.4k
1.8k
2.2k


Adam
Loss/raw
tag: Adam/Loss/raw
0.65
0.55
0.45
0.35
0.25
0
400
800
1.2k
1.6k
2k

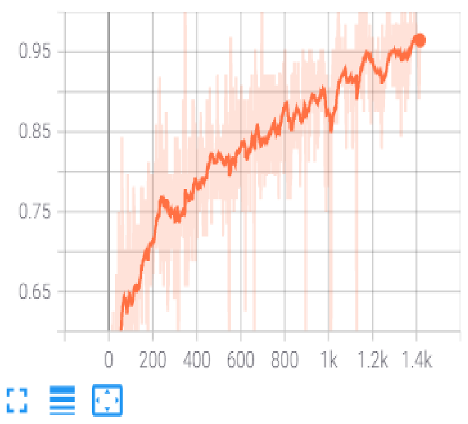
	Value	Train accuracy	Validation accuracy	Test accuracy
Image Size	150	96%	90%	77%
Epochs	30			
Learning Rate	0.001			
Color	Gray Scale			
Kernel Size	3			

Accuracy

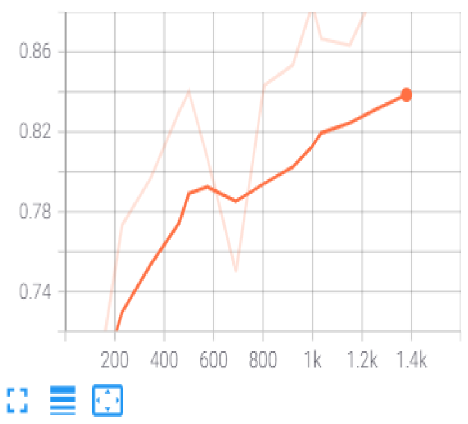
Accuracy



__raw__
tag: Accuracy/__raw__

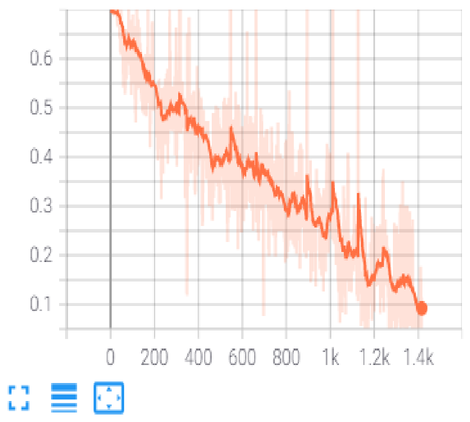


Validation
tag: Accuracy/Validation



Adam

Loss/raw
tag: Adam/Loss/raw



- Inception :-

  Hyperparameters :-

  - This model was compiled using momentum optimization algorithm, categorical cross-entropy as its loss function. The dropout layer used has a drop-out probability of 50%. The output layer has softmax as its activation function. The rest of the network uses ReLU.

Results :-

	Value	Train accuracy	Validation accuracy	Test accuracy
Image Size	224	95%	81%	73%
Epochs	30			
Learning Rate	0.001			
Color	RGB			
Kernel Size	3			