SOL PLAATJE UNIVERSITY

DETECTING AGE FROM IMAGES

Prepared by Kudzai Carlos Sibanda

201574939

NDEV

Data Science Problem

October 17, 2019

Contents

1 Problem Statement

Indian Movie Face database (IMFDB) is a large unconstrained face database consisting of 34512 images of 100 Indian actors collected from more than 100 videos. All the images are manually selected and cropped from the video frames resulting in a high degree of variability interms of scale, pose, expression, illumination, age, resolution, occlusion, and makeup. IMFDB is the first face database that provides a detailed annotation of every image in terms of age, pose, gender, expression and type of occlusion that may help other face related applications. The problem is we want to find out what age range a person belongs to based on a picture alone despite the image quality. This is a Data Science problem because it requires deep learning of the images in an attempt to find patterns or ways to classify them based on their attributes. Deep neural networks are built in order to do this as well as models and all this falls under data science. Such problems open up a whole new world of data analysis and push the boundaries of what we are able to do with deep learning and machine learning. Only a data scientist is able to perform the pre-processing and build models for this analysis. Extracting insights from data is what makes the data valuable. Data is useless if nothing can be gotten from it.

2 Maths

Mathematics is used in everything. In the pre-processing it is used to augment certain values, it is used to find the right dropout rate and the learning rate as well as normalizing the dataset. In this project i used convoluted neural networks and it is guided by mathemetics all through.Convoluted Neural networks take input and they go through layers where they are pooled and a loss function is applied to it as well as an activation function.

Without using activation functions the neural network would just be a combination of linear functions and it would just yield a linear result. The activation function also helps with the pace at which a model learns. In this project I used the reLu and the LeakyReLu as they were the ones best suited for my dataset and what i was trying to achieve.

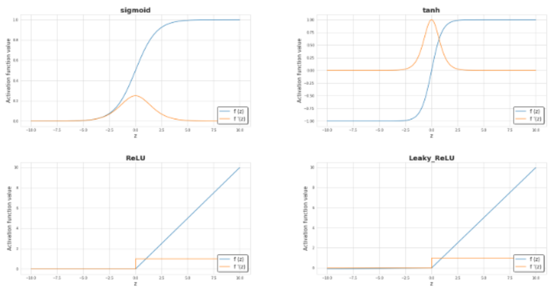


Figure 6. Diagrams of the most popular activation functions together with their derivatives.

The loss function shows you how far your model is from your ideal solution /model. With each iteration the loss value decreases as the accuracy increases. i used binary-crossentropy in my model.

$$J(W,b) = \frac{1}{m}\sum_{i=1}^{m} L(\hat{y}^{(i)}, y^{(i)})$$
$$L(\hat{y}, y) = -(y \log \hat{y} + (1-y)\log(1-\hat{y}))$$

Backpropagation is the algorithm that allows one to calculate the gradient. The parameters are adjustable based on this formula:

$$\mathbf{W}^{[l]} = \mathbf{W}^{[l]} - \alpha \mathbf{dW}^{[l]}$$
$$\mathbf{b}^{[l]} = \mathbf{b}^{[l]} - \alpha \mathbf{db}^{[l]}$$

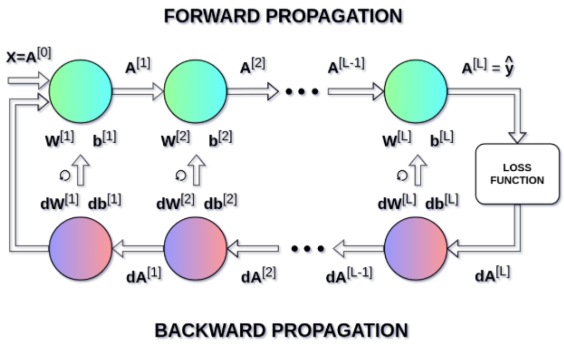
In the equations above,α represents learning rate- a hyperparameter which allows you to control the value of performed adjustment. Choosing a learning rate is crucial — we set it too low, our NN will be learning very slowly, we set it too high and we will not be able to hit the minimum. dW and db are calculated using the chain rule, partial derivatives of loss function with respect to W and b. The size of dW and db are the same as that of W and b respectively. The rest of backpropagation works as follows:

$$\mathbf{dW}^{[l]} = \frac{\partial L}{\partial \mathbf{W}^{[l]}} = \frac{1}{m}\mathbf{dZ}^{[l]}\mathbf{A}^{[l-1]T}$$

$$\mathbf{db}^{[l]} = \frac{\partial L}{\partial \mathbf{b}^{[l]}} = \frac{1}{m}\sum_{i=1}^{m}\mathbf{dZ}^{[l](i)}$$

$$\mathbf{dA}^{[l-1]} = \frac{\partial L}{\partial \mathbf{A}^{[l-1]}} = \mathbf{W}^{[l]T}\mathbf{dZ}^{[l]}$$

$$\mathbf{dZ}^{[l]} = \mathbf{dA}^{[l]} * g'(\mathbf{Z}^{[l]})$$



3 Data

The dataset is cleaned and formatted to give you a total of 26742 images with 19906 images in train and 6636 images in test.The task is to predict the age of a person from his or her facial attributes. For simplicity, the problem has been converted to a multiclass problem with classes as Young, Middle and Old.The attributes of data are as follows:

ID – Unique ID of image

Class – Age bin of person in image

The data was provided by Analyticsvidhya as part of a competition to build a model for age detection. It is two folders (test and train), which consists of thousands of images. I downloaded the data from their site

For this problem I employed the use of convoluted neural networks to try and classify the images into the three classes : young, middle aged and old ; based on only the ID of the images. The results were quite interesting and not what i expected at all.

4 Code Snippets

```
In [7]: #first 5 rows
        train.head()
```

Out[7]:

	ID	Class
0	377.jpg	MIDDLE
1	17814.jpg	YOUNG
2	21283.jpg	MIDDLE
3	16496.jpg	YOUNG
4	4487.jpg	MIDDLE

```
In [8]: #first 5 rows
        test.head()
```

Out[8]:

	ID
0	25321.jpg
1	989.jpg
2	19277.jpg
3	13093.jpg
4	5367.jpg

Figure 1: Testing and training data

I then tried to use auto encoders to see if the model could yield a higher accuracy rate because the first two models gave more or less the same accuracy level.

```
In [26]: model.summary()

Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
batch_normalization_1 (Batch (None, 32, 32, 3)         12        
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 30, 30, 32)        896       
_________________________________________________________________
batch_normalization_2 (Batch (None, 30, 30, 32)        128       
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 28, 28, 32)        9248      
_________________________________________________________________
batch_normalization_3 (Batch (None, 28, 28, 32)        128       
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 14, 14, 32)        0         
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 12, 12, 64)        18496     
_________________________________________________________________
batch_normalization_4 (Batch (None, 12, 12, 64)        256       
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 10, 10, 64)        36928     
_________________________________________________________________
batch_normalization_5 (Batch (None, 10, 10, 64)        256       
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 5, 5, 64)          0         
_________________________________________________________________
flatten_1 (Flatten)          (None, 1600)              0         
_________________________________________________________________
dropout_1 (Dropout)          (None, 1600)              0         
_________________________________________________________________
dense_1 (Dense)              (None, 384)               614784    
_________________________________________________________________
dropout_2 (Dropout)          (None, 384)               0         
_________________________________________________________________
dense_2 (Dense)              (None, 3)                 1155      
=================================================================
Total params: 682,287
Trainable params: 681,897
Non-trainable params: 390
_________________________________________________________________
```

Figure 2: Summary of first model

TRAINING WITHOUT DATA AUGMENTATION

In [33]:
```
# Training the model
model.fit(X_train, y_train, batch_size=32, epochs=2, validation_split=0.2)
```

```
4 - accuracy: 0.659 - ETA: 13s - loss: 0.7813 - accuracy: 0.659 - ETA: 12s - loss: 0.7813 - accuracy: 0.659 - ETA: 12s - los
s: 0.7810 - accuracy: 0.659 - ETA: 12s - loss: 0.7812 - accuracy: 0.659 - ETA: 12s - loss: 0.7811 - accuracy: 0.660 - ETA: 11
s - loss: 0.7810 - accuracy: 0.660 - ETA: 11s - loss: 0.7808 - accuracy: 0.660 - ETA: 11s - loss: 0.7810 - accuracy: 0.660 -
ETA: 10s - loss: 0.7809 - accuracy: 0.660 - ETA: 10s - loss: 0.7812 - accuracy: 0.660 - ETA: 10s - loss: 0.7808 - accuracy:
0.660 - ETA: 10s - loss: 0.7808 - accuracy: 0.660 - ETA: 9s - loss: 0.7807 - accuracy: 0.660 - ETA: 9s - loss: 0.7808 - accur
acy: 0.66 - ETA: 9s - loss: 0.7806 - accuracy: 0.66 - ETA: 9s - loss: 0.7810 - accuracy: 0.65 - ETA: 8s - loss: 0.7815 - accu
racy: 0.65 - ETA: 8s - loss: 0.7817 - accuracy: 0.65 - ETA: 8s - loss: 0.7819 - accuracy: 0.65 - ETA: 7s - loss: 0.7817 - acc
uracy: 0.65 - ETA: 7s - loss: 0.7814 - accuracy: 0.65 - ETA: 7s - loss: 0.7811 - accuracy: 0.65 - ETA: 7s - loss: 0.7811 - ac
curacy: 0.65 - ETA: 6s - loss: 0.7812 - accuracy: 0.65 - ETA: 6s - loss: 0.7812 - accuracy: 0.65 - ETA: 6s - loss: 0.7813 - a
ccuracy: 0.65 - ETA: 6s - loss: 0.7814 - accuracy: 0.65 - ETA: 5s - loss: 0.7815 - accuracy: 0.65 - ETA: 5s - loss: 0.7815 -
accuracy: 0.65 - ETA: 5s - loss: 0.7815 - accuracy: 0.65 - ETA: 5s - loss: 0.7815 - accuracy: 0.65 - ETA: 4s - loss: 0.7814 -
accuracy: 0.65 - ETA: 4s - loss: 0.7816 - accuracy: 0.65 - ETA: 4s - loss: 0.7812 - accuracy: 0.65 - ETA: 3s - loss: 0.7809 -
accuracy: 0.65 - ETA: 3s - loss: 0.7805 - accuracy: 0.65 - ETA: 3s - loss: 0.7803 - accuracy: 0.65 - ETA: 3s - loss: 0.7801 -
accuracy: 0.65 - ETA: 2s - loss: 0.7801 - accuracy: 0.65 - ETA: 2s - loss: 0.7798 - accuracy: 0.65 - ETA: 2s - loss: 0.7797 -
accuracy: 0.65 - ETA: 2s - loss: 0.7801 - accuracy: 0.65 - ETA: 1s - loss: 0.7804 - accuracy: 0.65 - ETA: 1s - loss: 0.7799 -
accuracy: 0.65 - ETA: 1s - loss: 0.7801 - accuracy: 0.65 - ETA: 0s - loss: 0.7797 - accuracy: 0.65 - ETA: 0s - loss: 0.7793 -
accuracy: 0.65 - ETA: 0s - loss: 0.7792 - accuracy: 0.65 - ETA: 0s - loss: 0.7790 - accuracy: 0.65 - 142s 9ms/step - loss: 0.
7789 - accuracy: 0.6593 - val_loss: 0.6910 - val_accuracy: 0.7072
```

Out[33]: `<keras.callbacks.callbacks.History at 0x2216c001dd8>`

Figure 3: accuracy and validation of first model with no data augmentation

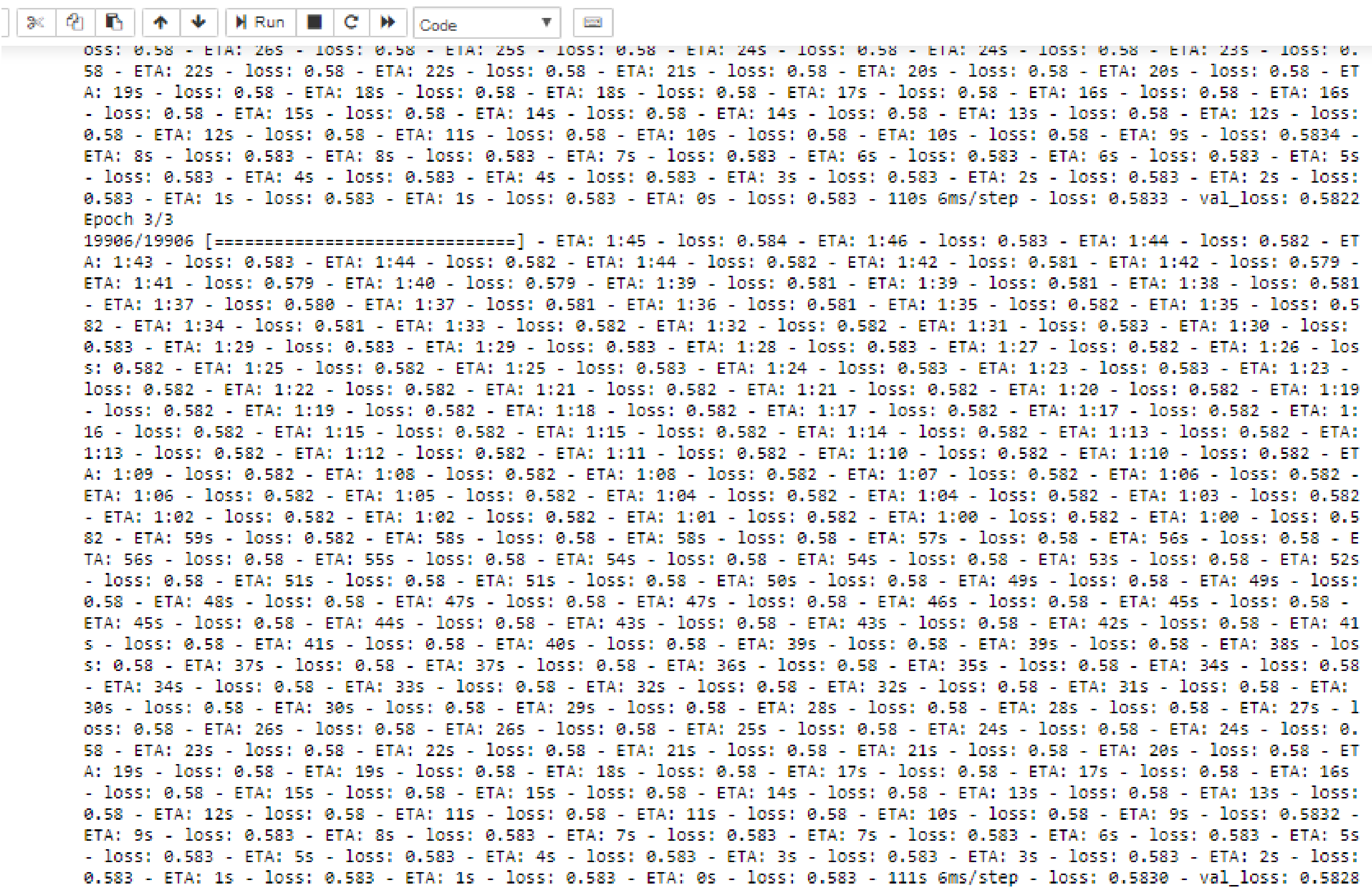
```
y: 0.65 - ETA: 1:04 - loss: 0.7824 - accuracy: 0.65 - ETA: 1:04 - loss: 0.7820 - accuracy: 0.65 - ETA: 1:03 - loss: 0.7822 - ac
curacy: 0.65 - ETA: 1:02 - loss: 0.7826 - accuracy: 0.65 - ETA: 1:02 - loss: 0.7825 - accuracy: 0.65 - ETA: 1:01 - loss: 0.7811
- accuracy: 0.65 - ETA: 1:01 - loss: 0.7793 - accuracy: 0.65 - ETA: 1:00 - loss: 0.7784 - accuracy: 0.65 - ETA: 59s - loss: 0.7
773 - accuracy: 0.6561 - ETA: 59s - loss: 0.7767 - accuracy: 0.656 - ETA: 58s - loss: 0.7769 - accuracy: 0.655 - ETA: 58s - los
s: 0.7778 - accuracy: 0.655 - ETA: 57s - loss: 0.7772 - accuracy: 0.656 - ETA: 56s - loss: 0.7774 - accuracy: 0.656 - ETA: 56s
- loss: 0.7781 - accuracy: 0.655 - ETA: 55s - loss: 0.7773 - accuracy: 0.656 - ETA: 55s - loss: 0.7769 - accuracy: 0.657 - ETA:
54s - loss: 0.7772 - accuracy: 0.657 - ETA: 53s - loss: 0.7767 - accuracy: 0.657 - ETA: 53s - loss: 0.7764 - accuracy: 0.657 -
ETA: 52s - loss: 0.7769 - accuracy: 0.657 - ETA: 52s - loss: 0.7772 - accuracy: 0.657 - ETA: 51s - loss: 0.7779 - accuracy: 0.6
56 - ETA: 50s - loss: 0.7783 - accuracy: 0.656 - ETA: 50s - loss: 0.7791 - accuracy: 0.656 - ETA: 49s - loss: 0.7787 - accurac
y: 0.656 - ETA: 49s - loss: 0.7789 - accuracy: 0.656 - ETA: 48s - loss: 0.7788 - accuracy: 0.656 - ETA: 48s - loss: 0.7784 - ac
curacy: 0.656 - ETA: 47s - loss: 0.7780 - accuracy: 0.656 - ETA: 46s - loss: 0.7790 - accuracy: 0.656 - ETA: 46s - loss: 0.7792
- accuracy: 0.656 - ETA: 45s - loss: 0.7795 - accuracy: 0.656 - ETA: 45s - loss: 0.7787 - accuracy: 0.656 - ETA: 44s - loss: 0.
7785 - accuracy: 0.656 - ETA: 44s - loss: 0.7783 - accuracy: 0.656 - ETA: 43s - loss: 0.7773 - accuracy: 0.657 - ETA: 43s - los
s: 0.7779 - accuracy: 0.657 - ETA: 42s - loss: 0.7780 - accuracy: 0.657 - ETA: 42s - loss: 0.7786 - accuracy: 0.657 - ETA: 41s
- loss: 0.7784 - accuracy: 0.657 - ETA: 41s - loss: 0.7785 - accuracy: 0.656 - ETA: 40s - loss: 0.7785 - accuracy: 0.656 - ETA:
39s - loss: 0.7782 - accuracy: 0.656 - ETA: 39s - loss: 0.7786 - accuracy: 0.656 - ETA: 38s - loss: 0.7791 - accuracy: 0.656 -
ETA: 38s - loss: 0.7797 - accuracy: 0.655 - ETA: 37s - loss: 0.7797 - accuracy: 0.655 - ETA: 37s - loss: 0.7787 - accuracy: 0.6
56 - ETA: 36s - loss: 0.7793 - accuracy: 0.656 - ETA: 36s - loss: 0.7795 - accuracy: 0.655248/248 [============================
==] - ETA: 35s - loss: 0.7795 - accuracy: 0.655 - ETA: 35s - loss: 0.7797 - accuracy: 0.655 - ETA: 34s - loss: 0.7797 - accurac
y: 0.655 - ETA: 33s - loss: 0.7790 - accuracy: 0.655 - ETA: 33s - loss: 0.7787 - accuracy: 0.656 - ETA: 32s - loss: 0.7778 - ac
curacy: 0.656 - ETA: 32s - loss: 0.7792 - accuracy: 0.655 - ETA: 31s - loss: 0.7790 - accuracy: 0.655 - ETA: 31s - loss: 0.7798
- accuracy: 0.655 - ETA: 30s - loss: 0.7794 - accuracy: 0.655 - ETA: 29s - loss: 0.7797 - accuracy: 0.655 - ETA: 29s - loss: 0.
7788 - accuracy: 0.656 - ETA: 28s - loss: 0.7789 - accuracy: 0.656 - ETA: 28s - loss: 0.7785 - accuracy: 0.656 - ETA: 27s - los
s: 0.7788 - accuracy: 0.656 - ETA: 27s - loss: 0.7784 - accuracy: 0.656 - ETA: 26s - loss: 0.7782 - accuracy: 0.656 - ETA: 26s
- loss: 0.7782 - accuracy: 0.656 - ETA: 25s - loss: 0.7789 - accuracy: 0.655 - ETA: 24s - loss: 0.7791 - accuracy: 0.655 - ETA:
24s - loss: 0.7787 - accuracy: 0.655 - ETA: 23s - loss: 0.7785 - accuracy: 0.655 - ETA: 23s - loss: 0.7787 - accuracy: 0.655 -
ETA: 22s - loss: 0.7785 - accuracy: 0.656 - ETA: 22s - loss: 0.7787 - accuracy: 0.656 - ETA: 21s - loss: 0.7783 - accuracy: 0.6
56 - ETA: 20s - loss: 0.7784 - accuracy: 0.656 - ETA: 20s - loss: 0.7776 - accuracy: 0.656 - ETA: 19s - loss: 0.7769 - accurac
y: 0.657 - ETA: 19s - loss: 0.7761 - accuracy: 0.657 - ETA: 18s - loss: 0.7753 - accuracy: 0.657 - ETA: 18s - loss: 0.7755 - ac
curacy: 0.657 - ETA: 17s - loss: 0.7747 - accuracy: 0.658 - ETA: 16s - loss: 0.7740 - accuracy: 0.658 - ETA: 16s - loss: 0.7741
- accuracy: 0.658 - ETA: 15s - loss: 0.7742 - accuracy: 0.658 - ETA: 15s - loss: 0.7738 - accuracy: 0.658 - ETA: 14s - loss: 0.
7731 - accuracy: 0.658 - ETA: 13s - loss: 0.7738 - accuracy: 0.658 - ETA: 13s - loss: 0.7735 - accuracy: 0.658 - ETA: 12s - los
s: 0.7730 - accuracy: 0.658 - ETA: 12s - loss: 0.7732 - accuracy: 0.658 - ETA: 11s - loss: 0.7731 - accuracy: 0.658 - ETA: 10s
- loss: 0.7728 - accuracy: 0.658 - ETA: 10s - loss: 0.7732 - accuracy: 0.657 - ETA: 9s - loss: 0.7724 - accuracy: 0.658 - ETA:
9s - loss: 0.7731 - accuracy: 0.65 - ETA: 8s - loss: 0.7728 - accuracy: 0.65 - ETA: 7s - loss: 0.7726 - accuracy: 0.65 - ETA: 7
s - loss: 0.7724 - accuracy: 0.65 - ETA: 6s - loss: 0.7721 - accuracy: 0.65 - ETA: 6s - loss: 0.7725 - accuracy: 0.65 - ETA: 5s
- loss: 0.7723 - accuracy: 0.65 - ETA: 4s - loss: 0.7730 - accuracy: 0.65 - ETA: 4s - loss: 0.7733 - accuracy: 0.65 - ETA: 3s -
loss: 0.7733 - accuracy: 0.65 - ETA: 3s - loss: 0.7733 - accuracy: 0.65 - ETA: 2s - loss: 0.7727 - accuracy: 0.65 - ETA: 1s - l
oss: 0.7720 - accuracy: 0.65 - ETA: 1s - loss: 0.7721 - accuracy: 0.65 - ETA: 0s - loss: 0.7719 - accuracy: 0.65 - 162s 654ms/s
tep - loss: 0.7715 - accuracy: 0.6585 - val_loss: 0.7133 - val_accuracy: 0.6757
```

Figure 4: accuracy and validation of first model with data augmentation

```
In [42]: #summary
         autoencoder.summary()
```

```
Model: "model_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         (None, 32, 32, 3)         0
_________________________________________________________________
conv2d_15 (Conv2D)           (None, 32, 32, 32)        896
_________________________________________________________________
max_pooling2d_9 (MaxPooling2 (None, 16, 16, 32)        0
_________________________________________________________________
conv2d_16 (Conv2D)           (None, 16, 16, 16)        4624
_________________________________________________________________
max_pooling2d_10 (MaxPooling (None, 8, 8, 16)          0
_________________________________________________________________
conv2d_17 (Conv2D)           (None, 8, 8, 8)           1160
_________________________________________________________________
max_pooling2d_11 (MaxPooling (None, 4, 4, 8)           0
_________________________________________________________________
conv2d_18 (Conv2D)           (None, 4, 4, 8)           584
_________________________________________________________________
up_sampling2d_1 (UpSampling2 (None, 8, 8, 8)           0
_________________________________________________________________
conv2d_19 (Conv2D)           (None, 8, 8, 16)          1168
_________________________________________________________________
up_sampling2d_2 (UpSampling2 (None, 16, 16, 16)        0
_________________________________________________________________
conv2d_20 (Conv2D)           (None, 16, 16, 32)        4640
_________________________________________________________________
up_sampling2d_3 (UpSampling2 (None, 32, 32, 32)        0
_________________________________________________________________
conv2d_21 (Conv2D)           (None, 32, 32, 3)         867
=================================================================
Total params: 13,939
Trainable params: 13,939
Non-trainable params: 0
_________________________________________________________________
```

Figure 5: auto encoder

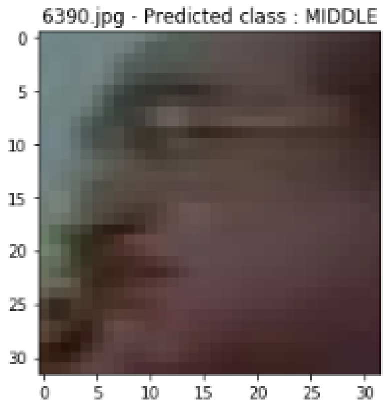



```
oss: 0.58 - ETA: 26s - loss: 0.58 - ETA: 25s - loss: 0.58 - ETA: 24s - loss: 0.58 - ETA: 24s - loss: 0.58 - ETA: 23s - loss: 0.
58 - ETA: 22s - loss: 0.58 - ETA: 22s - loss: 0.58 - ETA: 21s - loss: 0.58 - ETA: 20s - loss: 0.58 - ETA: 20s - loss: 0.58 - ET
A: 19s - loss: 0.58 - ETA: 18s - loss: 0.58 - ETA: 18s - loss: 0.58 - ETA: 17s - loss: 0.58 - ETA: 16s - loss: 0.58 - ETA: 16s
- loss: 0.58 - ETA: 15s - loss: 0.58 - ETA: 14s - loss: 0.58 - ETA: 14s - loss: 0.58 - ETA: 13s - loss: 0.58 - ETA: 12s - loss:
0.58 - ETA: 12s - loss: 0.58 - ETA: 11s - loss: 0.58 - ETA: 10s - loss: 0.58 - ETA: 10s - loss: 0.58 - ETA: 9s - loss: 0.5834 -
ETA: 8s - loss: 0.583 - ETA: 8s - loss: 0.583 - ETA: 7s - loss: 0.583 - ETA: 6s - loss: 0.583 - ETA: 6s - loss: 0.583 - ETA: 5s
- loss: 0.583 - ETA: 4s - loss: 0.583 - ETA: 4s - loss: 0.583 - ETA: 3s - loss: 0.583 - ETA: 2s - loss: 0.583 - ETA: 2s - loss:
0.583 - ETA: 1s - loss: 0.583 - ETA: 1s - loss: 0.583 - ETA: 0s - loss: 0.583 - 110s 6ms/step - loss: 0.5833 - val_loss: 0.5822
Epoch 3/3
19906/19906 [==============================] - ETA: 1:45 - loss: 0.584 - ETA: 1:46 - loss: 0.583 - ETA: 1:44 - loss: 0.582 - ET
A: 1:43 - loss: 0.583 - ETA: 1:44 - loss: 0.582 - ETA: 1:44 - loss: 0.582 - ETA: 1:42 - loss: 0.581 - ETA: 1:42 - loss: 0.579 -
ETA: 1:41 - loss: 0.579 - ETA: 1:40 - loss: 0.579 - ETA: 1:39 - loss: 0.581 - ETA: 1:39 - loss: 0.581 - ETA: 1:38 - loss: 0.581
- ETA: 1:37 - loss: 0.580 - ETA: 1:37 - loss: 0.581 - ETA: 1:36 - loss: 0.581 - ETA: 1:35 - loss: 0.582 - ETA: 1:35 - loss: 0.5
82 - ETA: 1:34 - loss: 0.581 - ETA: 1:33 - loss: 0.582 - ETA: 1:32 - loss: 0.582 - ETA: 1:31 - loss: 0.583 - ETA: 1:30 - loss:
0.583 - ETA: 1:29 - loss: 0.583 - ETA: 1:29 - loss: 0.583 - ETA: 1:28 - loss: 0.583 - ETA: 1:27 - loss: 0.582 - ETA: 1:26 - los
s: 0.582 - ETA: 1:25 - loss: 0.582 - ETA: 1:25 - loss: 0.583 - ETA: 1:24 - loss: 0.583 - ETA: 1:23 - loss: 0.583 - ETA: 1:23 -
loss: 0.582 - ETA: 1:22 - loss: 0.582 - ETA: 1:21 - loss: 0.582 - ETA: 1:21 - loss: 0.582 - ETA: 1:20 - loss: 0.582 - ETA: 1:19
- loss: 0.582 - ETA: 1:19 - loss: 0.582 - ETA: 1:18 - loss: 0.582 - ETA: 1:17 - loss: 0.582 - ETA: 1:17 - loss: 0.582 - ETA: 1:
16 - loss: 0.582 - ETA: 1:15 - loss: 0.582 - ETA: 1:15 - loss: 0.582 - ETA: 1:14 - loss: 0.582 - ETA: 1:13 - loss: 0.582 - ETA:
1:13 - loss: 0.582 - ETA: 1:12 - loss: 0.582 - ETA: 1:11 - loss: 0.582 - ETA: 1:10 - loss: 0.582 - ETA: 1:10 - loss: 0.582 - ET
A: 1:09 - loss: 0.582 - ETA: 1:08 - loss: 0.582 - ETA: 1:08 - loss: 0.582 - ETA: 1:07 - loss: 0.582 - ETA: 1:06 - loss: 0.582 -
ETA: 1:06 - loss: 0.582 - ETA: 1:05 - loss: 0.582 - ETA: 1:04 - loss: 0.582 - ETA: 1:04 - loss: 0.582 - ETA: 1:03 - loss: 0.582
- ETA: 1:02 - loss: 0.582 - ETA: 1:02 - loss: 0.582 - ETA: 1:01 - loss: 0.582 - ETA: 1:00 - loss: 0.582 - ETA: 1:00 - loss: 0.5
82 - ETA: 59s - loss: 0.582 - ETA: 58s - loss: 0.58 - ETA: 58s - loss: 0.58 - ETA: 57s - loss: 0.58 - ETA: 56s - loss: 0.58 - E
TA: 56s - loss: 0.58 - ETA: 55s - loss: 0.58 - ETA: 54s - loss: 0.58 - ETA: 54s - loss: 0.58 - ETA: 53s - loss: 0.58 - ETA: 52s
- loss: 0.58 - ETA: 51s - loss: 0.58 - ETA: 51s - loss: 0.58 - ETA: 50s - loss: 0.58 - ETA: 49s - loss: 0.58 - ETA: 49s - loss:
0.58 - ETA: 48s - loss: 0.58 - ETA: 47s - loss: 0.58 - ETA: 47s - loss: 0.58 - ETA: 46s - loss: 0.58 - ETA: 45s - loss: 0.58 -
ETA: 45s - loss: 0.58 - ETA: 44s - loss: 0.58 - ETA: 43s - loss: 0.58 - ETA: 43s - loss: 0.58 - ETA: 42s - loss: 0.58 - ETA: 41
s - loss: 0.58 - ETA: 41s - loss: 0.58 - ETA: 40s - loss: 0.58 - ETA: 39s - loss: 0.58 - ETA: 39s - loss: 0.58 - ETA: 38s - los
s: 0.58 - ETA: 37s - loss: 0.58 - ETA: 37s - loss: 0.58 - ETA: 36s - loss: 0.58 - ETA: 35s - loss: 0.58 - ETA: 34s - loss: 0.58
- ETA: 34s - loss: 0.58 - ETA: 33s - loss: 0.58 - ETA: 32s - loss: 0.58 - ETA: 32s - loss: 0.58 - ETA: 31s - loss: 0.58 - ETA:
30s - loss: 0.58 - ETA: 30s - loss: 0.58 - ETA: 29s - loss: 0.58 - ETA: 28s - loss: 0.58 - ETA: 28s - loss: 0.58 - ETA: 27s - l
oss: 0.58 - ETA: 26s - loss: 0.58 - ETA: 26s - loss: 0.58 - ETA: 25s - loss: 0.58 - ETA: 24s - loss: 0.58 - ETA: 24s - loss: 0.
58 - ETA: 23s - loss: 0.58 - ETA: 22s - loss: 0.58 - ETA: 21s - loss: 0.58 - ETA: 21s - loss: 0.58 - ETA: 20s - loss: 0.58 - ET
A: 19s - loss: 0.58 - ETA: 19s - loss: 0.58 - ETA: 18s - loss: 0.58 - ETA: 17s - loss: 0.58 - ETA: 17s - loss: 0.58 - ETA: 16s
- loss: 0.58 - ETA: 15s - loss: 0.58 - ETA: 15s - loss: 0.58 - ETA: 14s - loss: 0.58 - ETA: 13s - loss: 0.58 - ETA: 13s - loss:
0.58 - ETA: 12s - loss: 0.58 - ETA: 11s - loss: 0.58 - ETA: 11s - loss: 0.58 - ETA: 10s - loss: 0.58 - ETA: 9s - loss: 0.5832 -
ETA: 9s - loss: 0.583 - ETA: 8s - loss: 0.583 - ETA: 7s - loss: 0.583 - ETA: 7s - loss: 0.583 - ETA: 6s - loss: 0.583 - ETA: 5s
- loss: 0.583 - ETA: 5s - loss: 0.583 - ETA: 4s - loss: 0.583 - ETA: 3s - loss: 0.583 - ETA: 3s - loss: 0.583 - ETA: 2s - loss:
0.583 - ETA: 1s - loss: 0.583 - ETA: 1s - loss: 0.583 - ETA: 0s - loss: 0.583 - 111s 6ms/step - loss: 0.5830 - val_loss: 0.5828
```

Figure 6: auto encoder results..significant drop in the loss

In [51]:
```
#testing model
i = np.random.choice(np.arange(len(test_data)))
plt.title('{} - Predicted class : {}'.format(test['ID'].values[i], pred_labels[i]))
plt.imshow(test_data[i])
```

Out[51]: <matplotlib.image.AxesImage at 0x22103b11f98>



In [52]:
```
#save model
subm = pd.DataFrame({'Class':pred_labels, 'ID':test.ID})
subm.to_csv('predictions.csv', index=False)
```

In [53]:
```
model.save("model.h5")
```

Figure 7: testing model against random images