Identify the dance form

# Architecture Used

Input layer (400 x 400 x 3)

Inception V3 with pre-trained weights (mixed7 layer)

Convolution layer with 150 filters 1 x 1

Convolution layers with 150 filters 2 x 2

Max Pooling 2 x 2

Dense 300 (relu)

Dropout 0.5

Dense 75 (relu)

Dense 8 (softmax)

A detailed architecture image is also shared in the zip file

## Using the model’s prediction for evaluation purpose

A file named final\_model is shared that will load the model with its weights and can be used to predict any images further for the evaluation purpose.

Just you need to put model.h5 file in your google drive in “save model 2” folder and run the final\_model file on google colab

## Note:

The model will predict the one-hot encoding which can be decoded using following table

classification=['bharatanatyam','kathak','kathakali','kuchipudi','manipuri','mohiniyattam','odissi','sattriya']

## Training Details

The model is designed in such a way to prevent it from overfitting and thus can be trained longer. I have trained my model for nearly 400 epoch on the whole data set with a learning rate of 0.0002 with rmsprop optimizer. I have used a Batch size of 128.

For training, you need to use a train generator and thus you need to upload the Train folder shared in zip to google drive. It should be placed in drive at stated path: “testing/train”

## Callbacks

Callbacks are also being used while training to save the model without interrupting the learning process. As we don’t have a labeled test set to check the performance, it is necessary to do so.

## Data Augmentation

Tensorflow has inbuilt data augmentation in Image generator. Using that I randomly tilted, flipped, sheared the images.

This is an important feature as it can prevent overfitting, given the dataset is small. And it can also make the model learn more efficiently

## Converging to the final architecture

For training any machine learning model we need labeled test data which was missing in this case hence I split the given training data set into two parts 284 labeled training set and 80 labeled training set, which is 70-30 percent nearly. So now I have 284 training data and 80 validation data.

I have used TensorFlow for training my model. Tensorflow has a feature of Image generator which can generate images for model.fit in batches. It trains the model on labeled training data and gives the performance measure on validation data at the same time.

Thus according to the labels given, I arranged my training images folderwise. You can find the same in the zip file.

Now I only have 284 labeled images and the training on them directly will definitely lead to overfitting. So just for the comparison I first tried to have some couple of Convolution layers directly on the image. This couple of layers lead me to the accuracy of 50% on validation data.

As we have a small data I decided to use transfer learning. For this, I used a built-in Inceptions model V3 with pre-trained weights of imagenet.

This could lead us to detect edges and important features very easily without being having a larger data set.

On this transfer learning model, I just choose the last layer and put some convolution layers on it. After training for 100 epoch my accuracy on validation data jumped to 70%+.

With some optimization, it leads me to 75 - 80% accuracy

After lots and lots of trials with architecture and hyperparameter, I finally decided to use mixed7 layer of inception model and got the F1 score of 83 on the test data just by training on 284 images. (validation data was not included yet)

Later I trained the same architecture with whole training data. I was expecting to have an F1 score of 90+ but even after training on the whole dataset, I was only able to get an F1 score of 87. This was probably because we were overfitting.

Without validation data, I can’t get an idea when the model is overfitting, so the only possible way was to save my model after each epoch using callbacks. At the end I tried to predict the test images with some of the best models I made.

This approach helpe me to overcome overfitting without having validation data.

After fine-tuning my model, I was able to get nearly an F1 score of 92.