

ISY5002 PATTERN RECOGNITION SYSTEMS

Group Project Report

**GAN based Fashion garment generator**

NACHIKETH DORAISWAMY (A0215523N)

LAKSHMI SUBRAMANIAN (A0215255L)

YALAVARTI DHARMA TEJA (A0215457A)

**Contents**

[1. Introduction 4](#_Toc54424781)

[2. Tools/Technique Used 4](#_Toc54424782)

[Tools used 4](#_Toc54424783)

[System Design / Models 5](#_Toc54424784)

[System Design 5](#_Toc54424785)

[Dataset 5](#_Toc54424786)

[Image Classification 5](#_Toc54424787)

[Pattern Classification 5](#_Toc54424788)

[System Performance 6](#_Toc54424789)

[Findings and Discussions 6](#_Toc54424790)

# Introduction

# Tools/Technique Used

## Tools used

* Jupyter Notebook/Colab
* Pycharm
* VS Code

Packages Used

* Tensorflow/Keras
* CV2
* Pytorch
* Scikit

Techniques Used

* Convolutional Neural Network
* ResNet
* Inception Resnet V2
* Stacked Ensemble model
* GAN
* DCGAN
* Color Detection using OpenCV

# 

# System Design / Models

## System Design

## Dataset

The dataset contains around 15k images with details of dress patters. The dataset was taken from the following page <https://s3.eu-central-1.amazonaws.com/fashion-gan/images.zip>.

Our project consists of 2 modules, the first being the classification of clothes based on pattern on the dress and the second is the generation of garments.

## Image Classification

In the first module We have developed classifier model to detect various attributes of a given image. The following are the few attributes we obtain from the image –

1. Pattern
2. Sleeve Length
3. Length
4. Color
5. Fit
6. Neckline

Pattern Classification

There are 6 output classes for the design patterns of the clothes. We developed multiple CNN models as base models over which a stacked ensemble model is built for classification. The distribution of data in the class is ununiform.

|  |  |
| --- | --- |
| **Class** | **No. of image data** |
| Floral | 2591 |
| Lace | 1018 |
| Polkadots | 428 |
| Print | 1220 |
| Stripes | 1010 |
| Unicolors | 6098 |

### Base Model-1

We have trained the dataset on a sequential Convolutional neural network with the following architecture, and observed a validation accuracy of 78.01%.

The hyper parameters set were RMSprop optimizer with a learning rate of 0.0001, there are many dropout layers with 25% rate and 50% in one of the last layers. Two convolution layers before the dense layers were set to L2 kernel regularization of penalty 0.001.

The following is the plot for training and test set loss and accuracy over 50 epochs.

The architecture and results of individual base models are as follows.

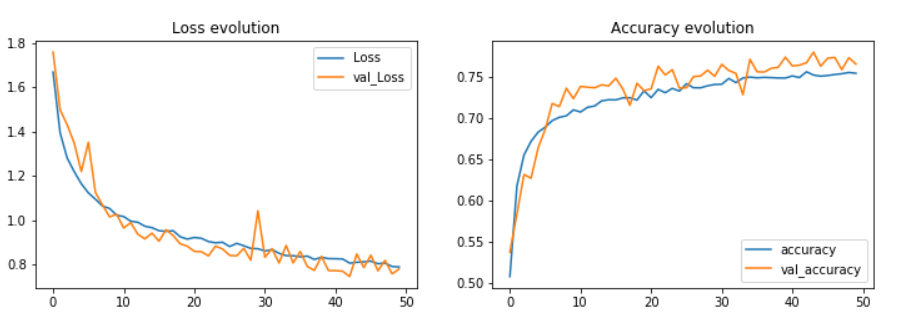


Fig – Base Model 1 – loss and accuracy evolution over 50 epochs.

The following is the confusion matrix for the above trained model.

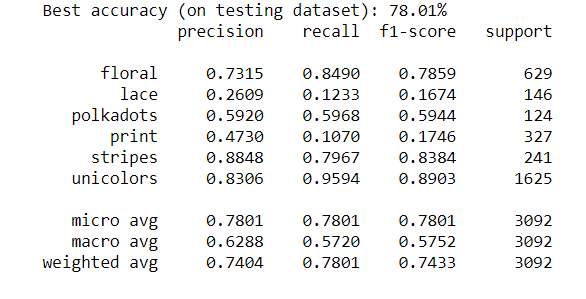


Fig – Base Model 1 – Confusion Matrix

### Base Model-2

The idea behind using the second CNN architecture was to test the accuracy using skip connections inspired from concept of resnet. Here the same dataset was trained using a slightly different architecture and hyperparameters. There are 7 convolution layers and 2 skip connections with RMSProp optimizer and learnig rate of 0.0001. Layers of BatchNorm, Activation(relu), Dropouts and MaxPooling were used. The model was trained for nearly 25 epochs.

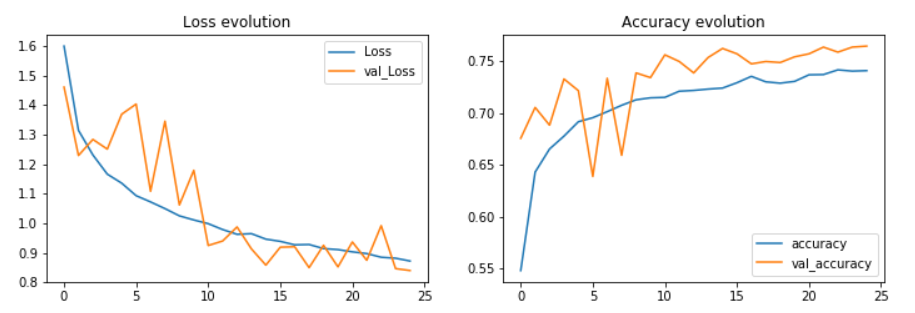


Fig – Base Model 2 – loss and accuracy evolution over 25 epochs.

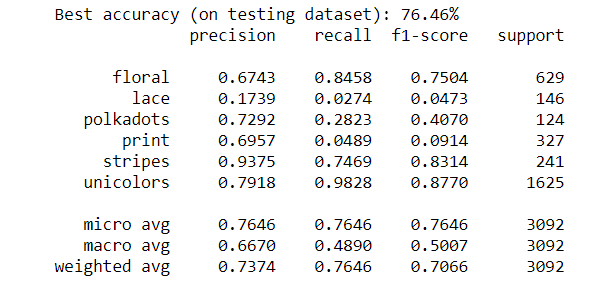


Fig – Base Model 2 – Confusion Matrix

### Base Model-3

The third CNN model used in ensemble model is Residual Networks, we have used 3 residual blocks containing 3 sub-blocks of multiple residual layers with the setting of Adam optimizer and he\_normal for initializing weight matrix. There are no Maxpooling and Dropout layers, only layers of Convolution, BatchNorm, Activation and AveragePooling was used in the architecture. This model was trained on 4000 images (30 epochs) and 15K images (10 epochs) separately and showed similar results with an accuracy of around 76.58% and 75.16% respectively. The architecture and results are shown below.

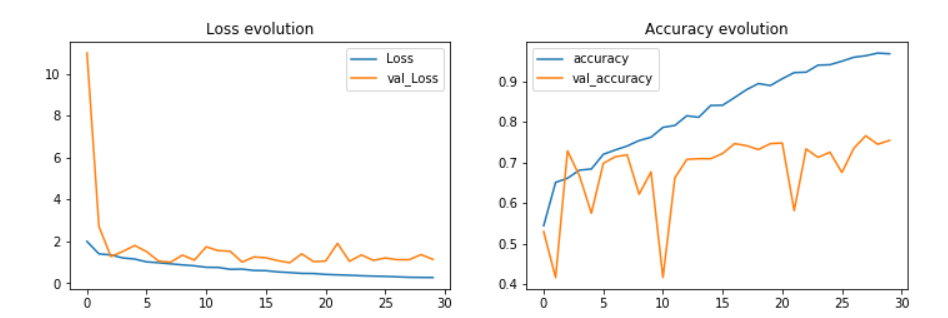


Fig – Base Model 3 – loss and accuracy evolution over 30 epochs with dataset of 4000 images.

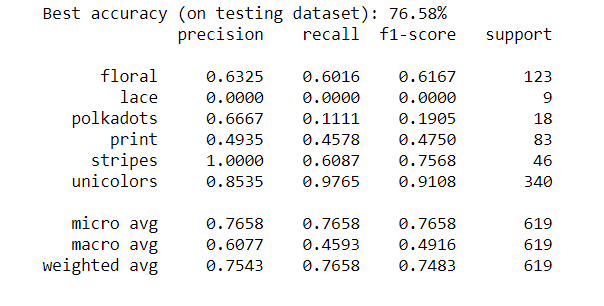


Fig – Base Model 3 – loss and accuracy evolution over 30 epochs with dataset of 4000 images.

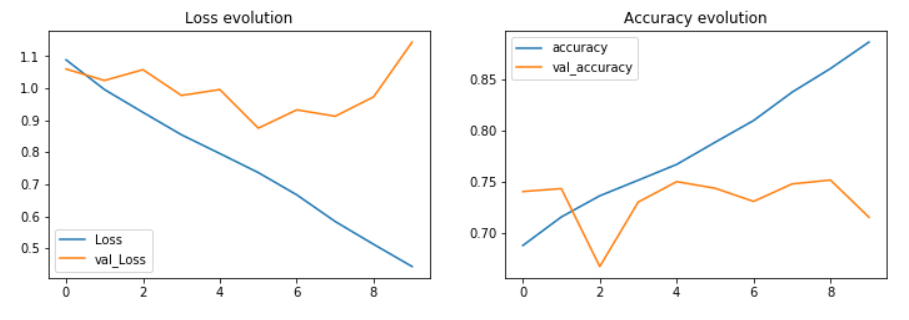


Fig – Base Model 3 – loss and accuracy evolution over 30 epochs with dataset of 15k images.

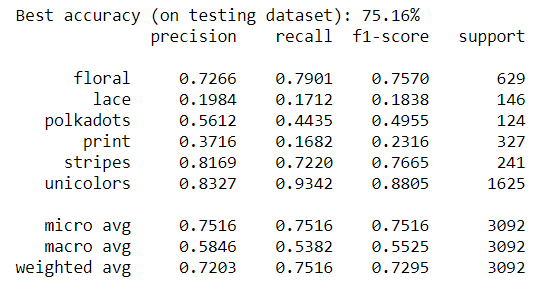
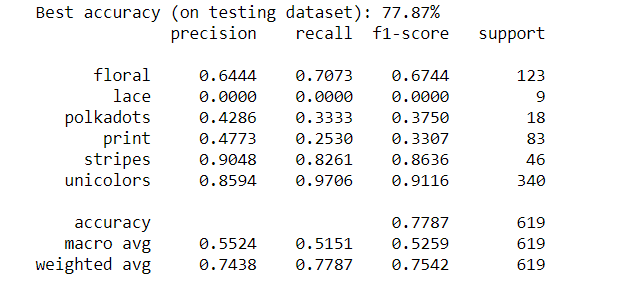
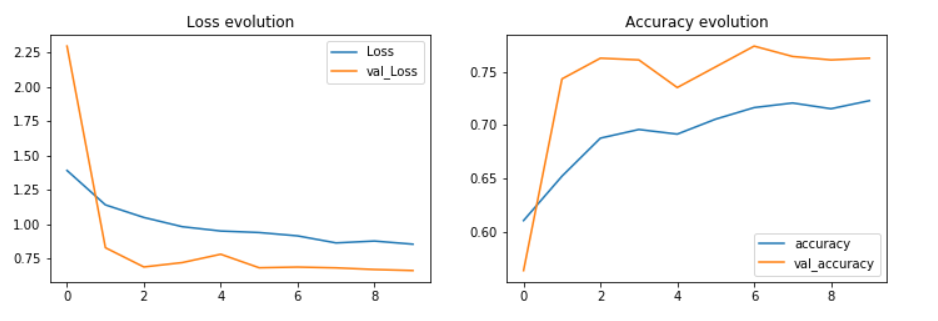


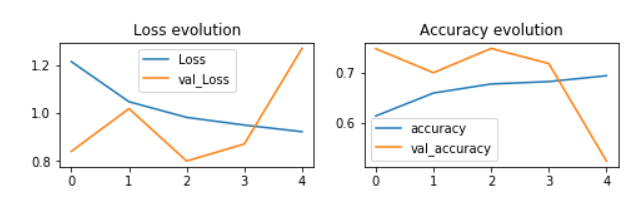
Fig – Base Model 3 – loss and accuracy evolution over 30 epochs with dataset of 15k images.

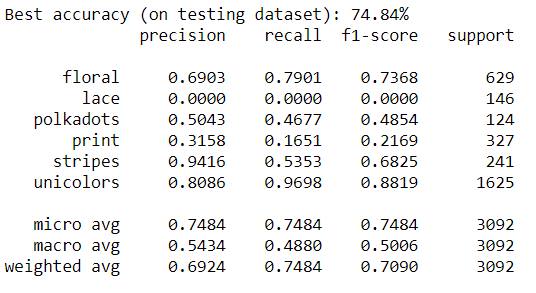
### Base Model-4

The fourth model used for ensemble is a simpler version of InceptionResNet V2, where only Inception ResNet-A was used for training due to system constraints.



Original results for the same:

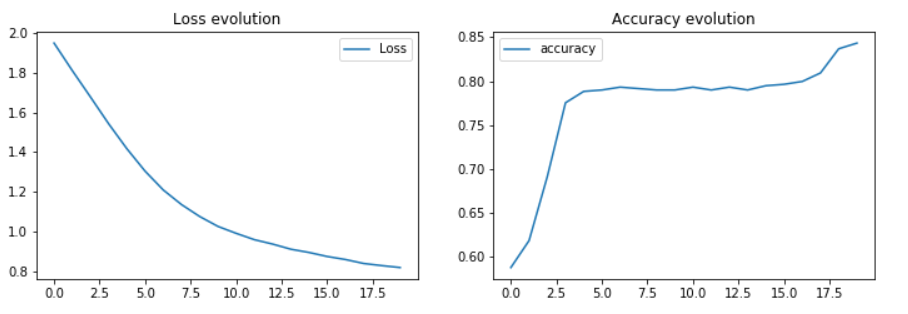


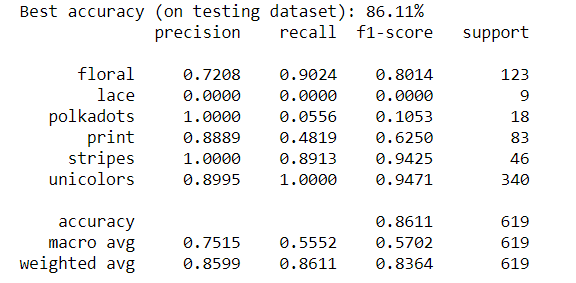


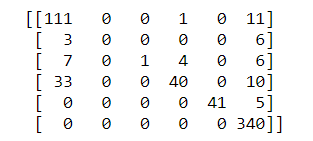
### Base Model-5

Nachiketh’s CNN model

### Ensemble model

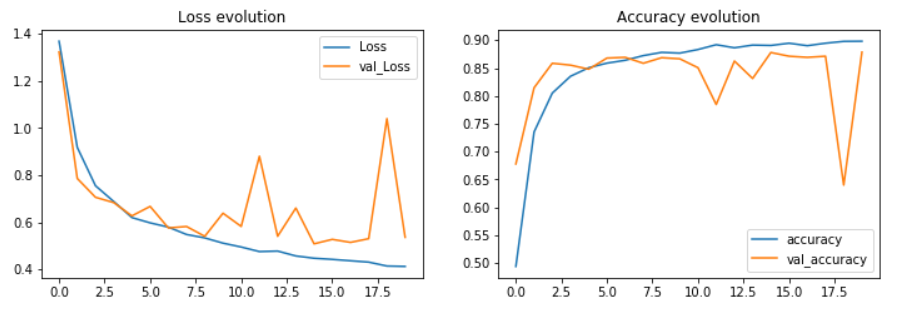


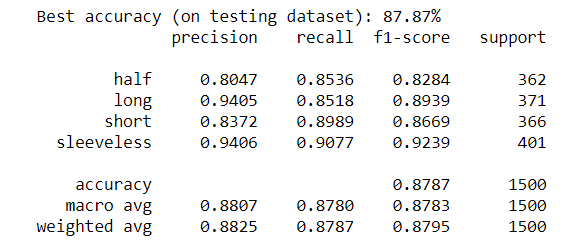




Sleevelength Prediction using CNN

There are 4 classes, the dataset used for training the CNN is around 1500 images per class.





Sleeve Length Classification

The length of the sleeves o

System Performance

Findings and Discussions

Oo

K

We have developed a few base models of Convolutional Neural network for classification. I had tried simple Sequential CNN models with multiple layers of Conv2D, BatchNorm, Activation and MaxPooling2D. I had also explored on many ConvNets like Inception, ResNet, and DenseNet.

Pattern Prediction using Stacked Ensemble

Appendix

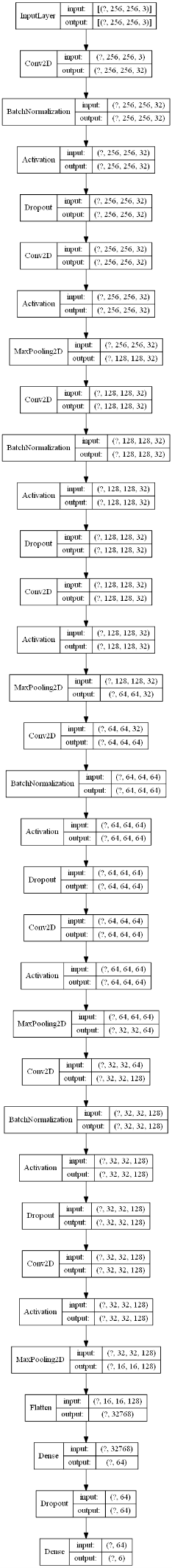
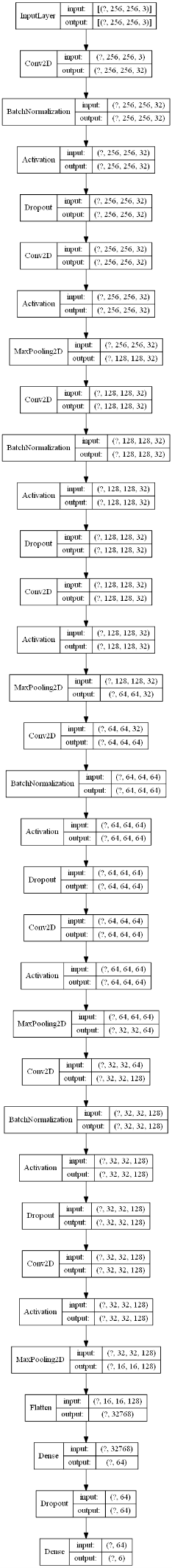


Fig – Base Model 1 – CNN Pattern Prediction Architecture

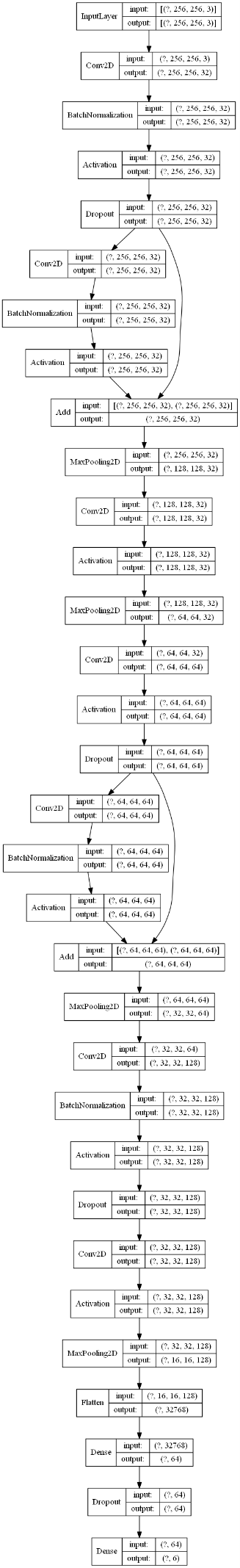
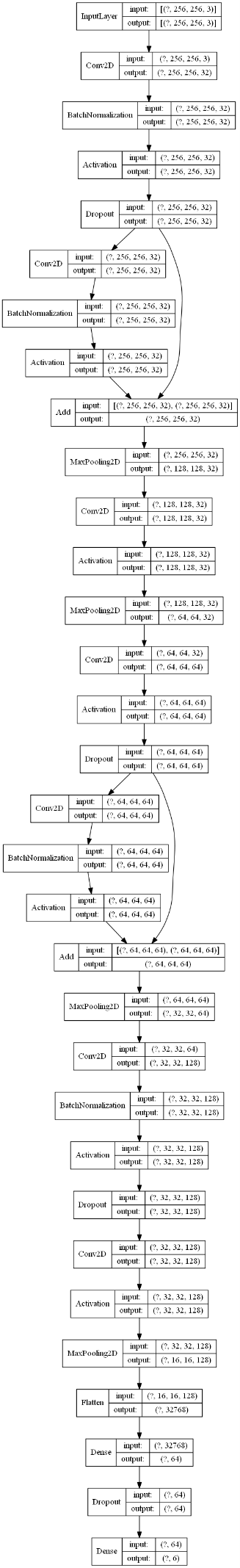
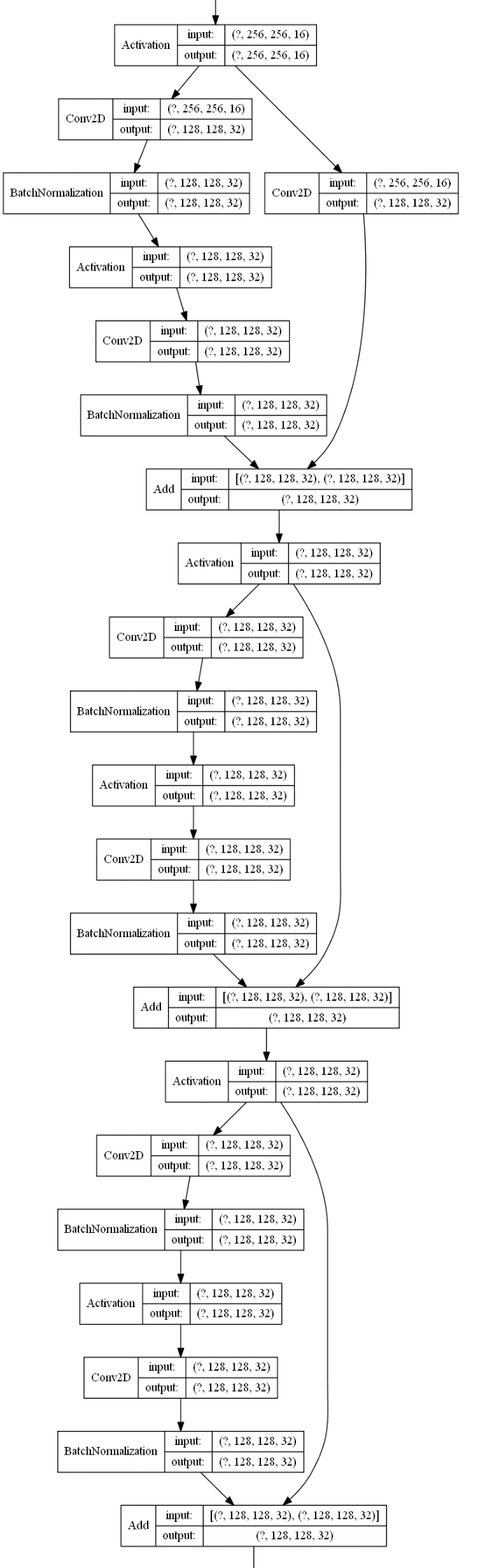
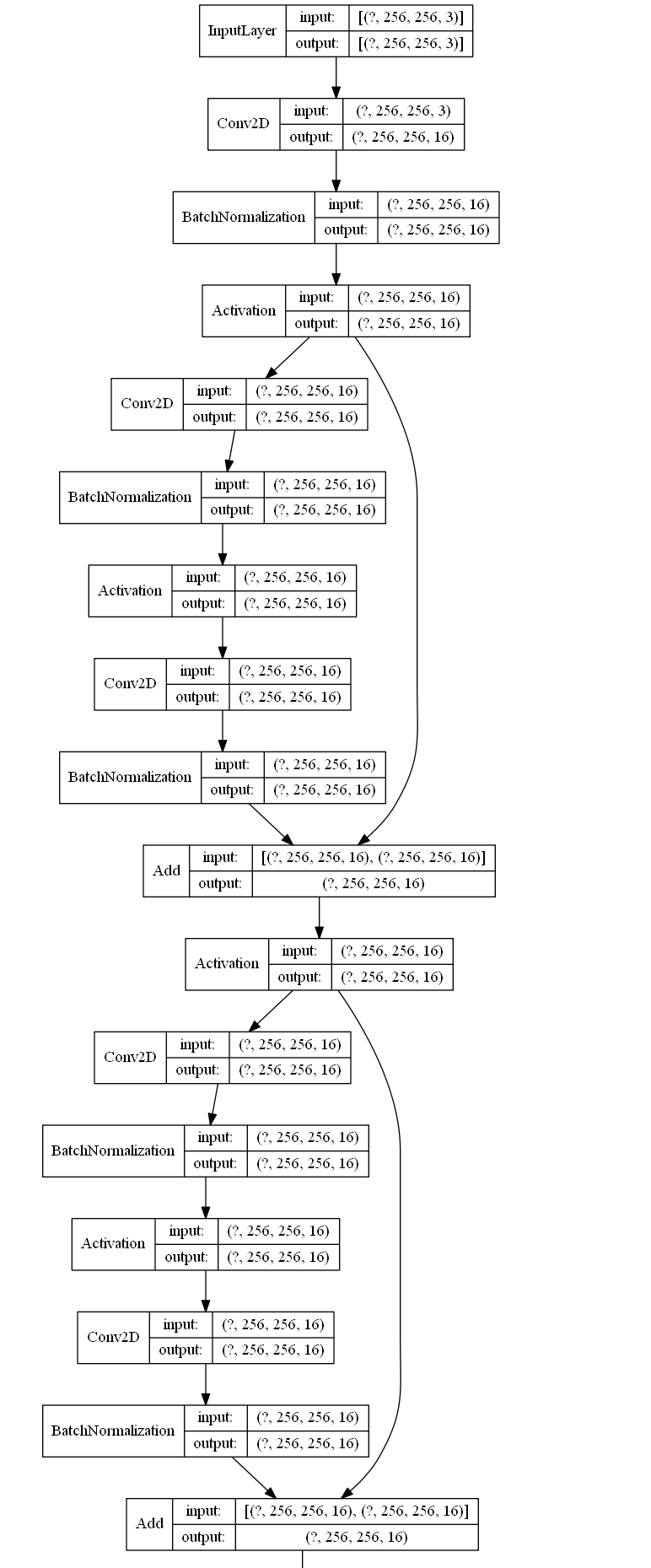


Fig – Base Model 2 – CNN Pattern Prediction Architecture



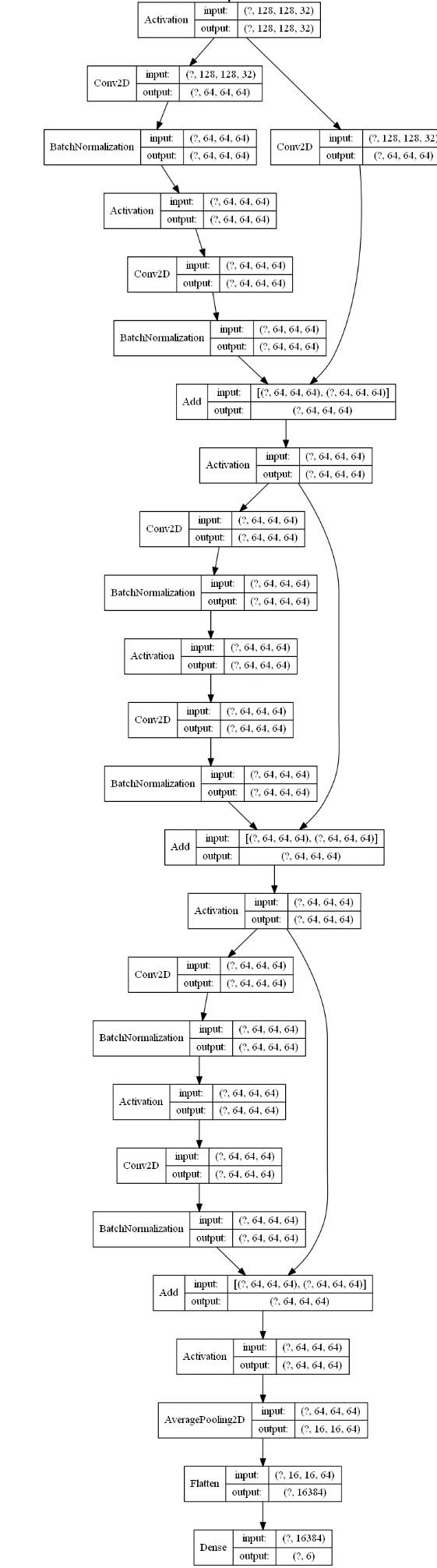


Fig – Base Model 3 – Pattern Prediction Architecture Resnet

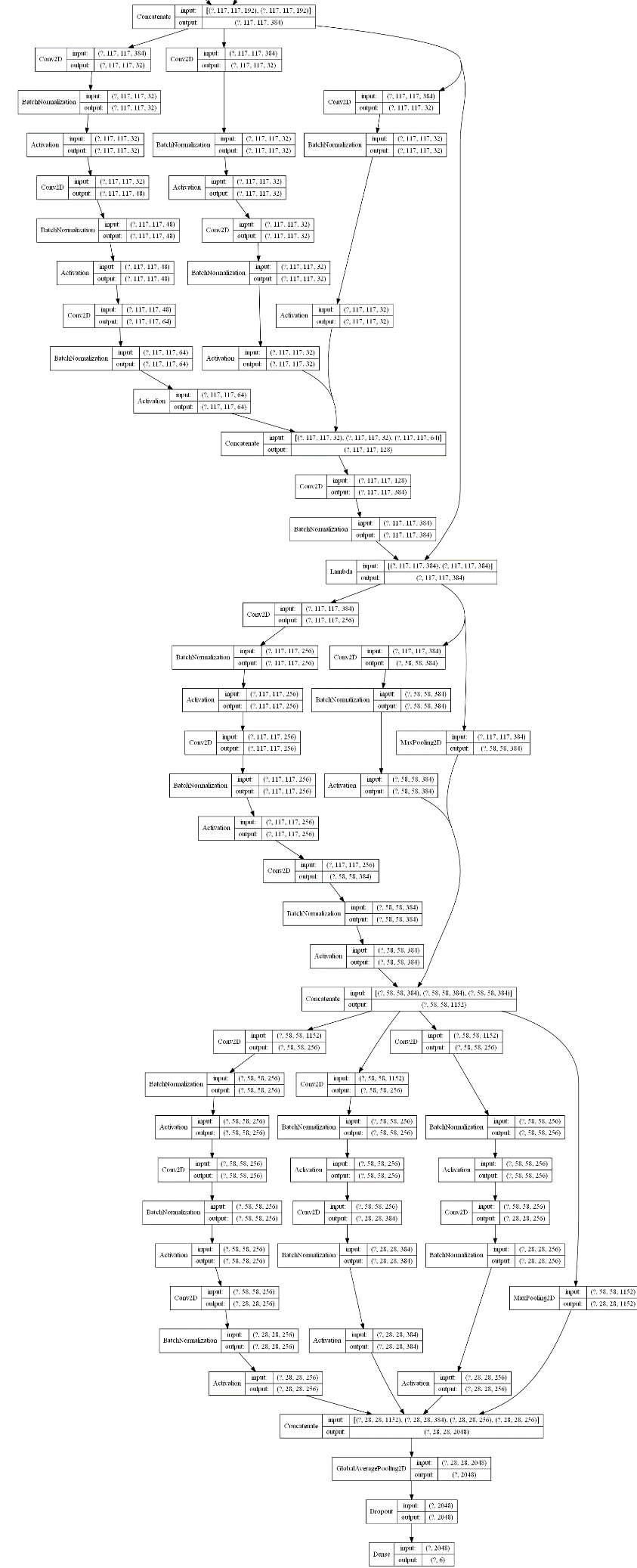
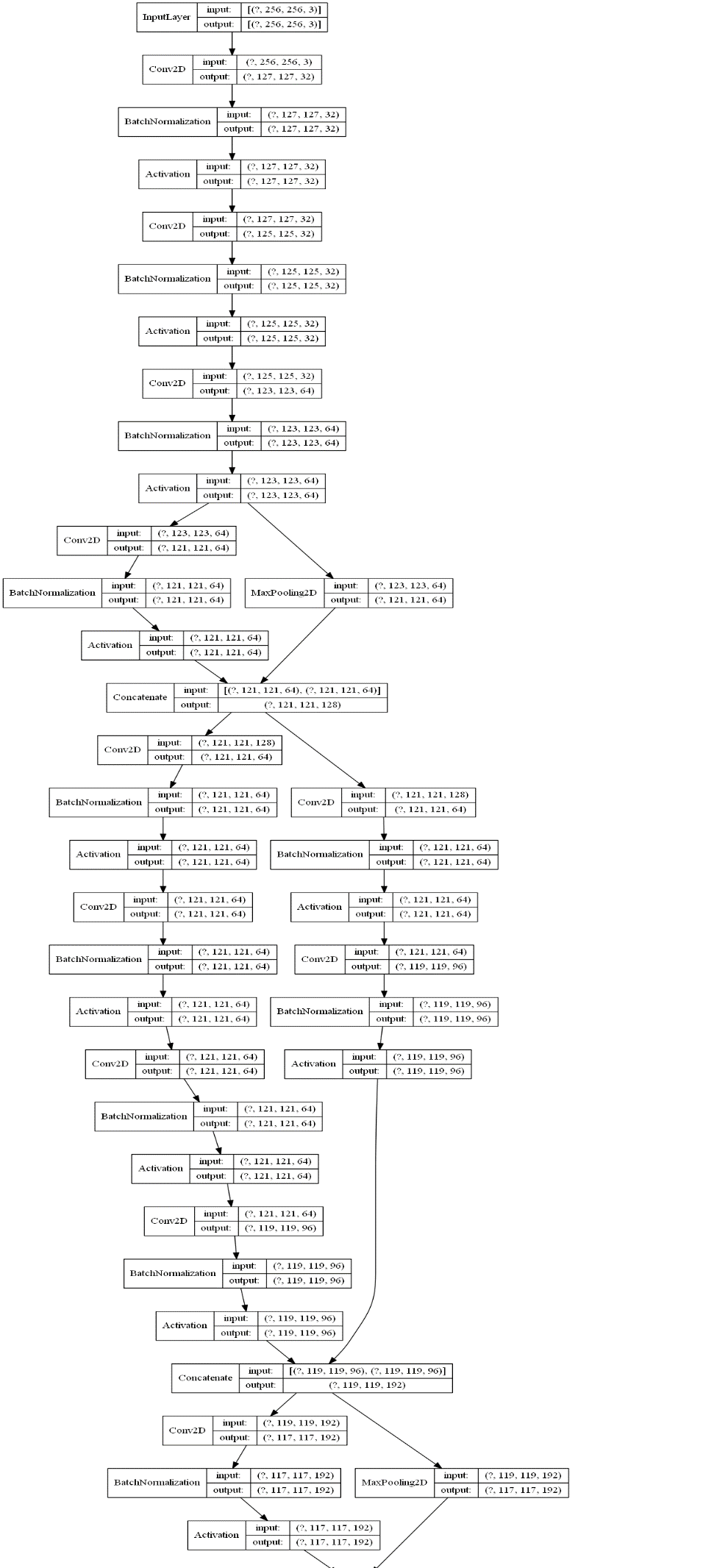


Fig – Base Model 4 – Pattern Prediction Architecture Inception Resnet V2

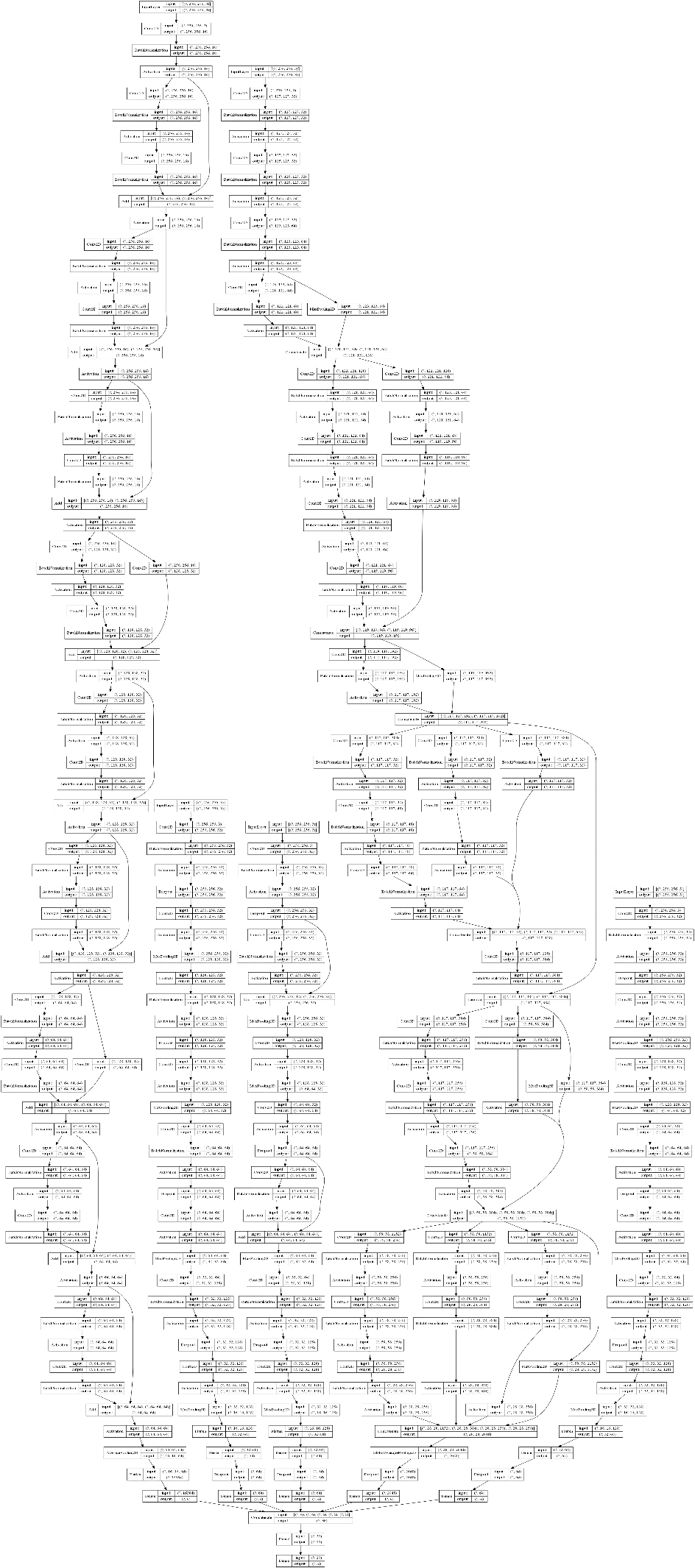


Fig – 6 Pattern Prediction Stacked Ensemble Model

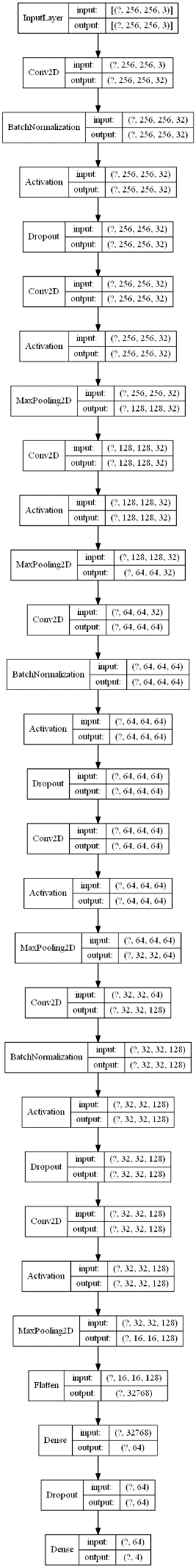
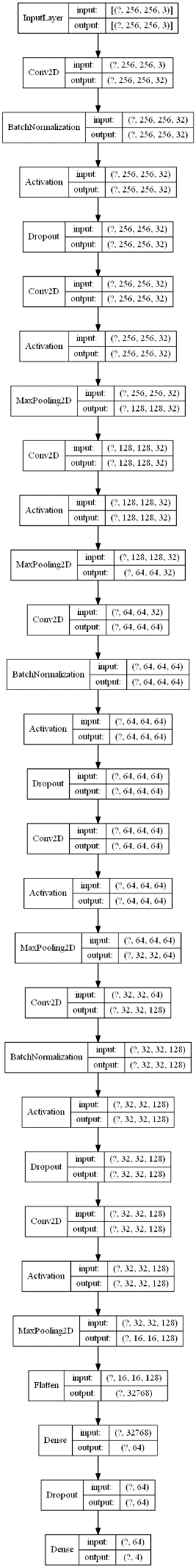


Fig – 7 – CNN Architecture for Sleeve length Prediction

Bibliography

<https://medium.com/the-owl/building-inception-resnet-v2-in-keras-from-scratch-a3546c4d93f0> inception resnet code from