PedVis-VGG-16: A Fine-tuned deep convolutional neural network for pedestrian image classifications

Fardin Hossain (1705038)
Pushpita Joardar (1705052)

Problem Definition

Predict if pedestrian(s) exist in an image.

No Pedestrian

Pedestrian

Dataset

- It is a combined dataset.
- It is mostly based on **INRIA Person Dataset.**

References:

https://github.com/vinay0410/Pedestrian_Detection **(neg: 1218,pos: 2416)**

https://www.kaggle.com/datasets/constantinwerner/human-detection-dataset **(neg: 362,pos:559)**

https://github.com/mahavird/Person-Detector-Dataset/tree/master/images **(pos: 200)**

https://github.com/RashadGarayev/PersonDetection.git **(neg:4000,pos:2000)**

Paper: https://drive.google.com/file/d/1QHTAheNuabj_5Vb7G2oSG6DsXUcdew96/view?usp=sharing

Dataset Analysis

PedVis Architecture:

Combined Dataset
Number of images for training : **9000**
Validation : **1800 (20%)**
Number of classes : **2 (Negative and Positive)**
Negative images: **5220**
Positive images: **3780**

Number of images for testing : **1900**
Number of classes: **2 (Negative and Positive)**
Negative images: **1102**
Positive images: **798**
Image Ratio in both sectors : **58:42(negative:positive)**

Proposed solution (architecture) : PedVis VGG16

	Layer	Output Shape	parameters
Block 1	Conv2D	[224, 128,64]	1792
	Conv2D	[224, 128, 64]	36928
	MaxPooling2D	[112, 64, 64]	0
Block 2	Conv2D	[112, 64, 128]	73856
	Conv2D	[112, 64, 128]	147584
	MaxPooling 2D	[56, 32, 128]	0
Block 3	Conv2D	[56, 32, 256]	295168
	Conv2D	[56,32,256]	590080
	Conv2D	[56,32,256]	590080
	MaxPooling 2D	[28,16,256]	0
Block 4	Conv2D	[28,16,512]	1188160
	Conv2D	[28,16, 512]	2359808
	Conv2D	[28 ,16, 512]	2359808
	MaxPooling 2D	[14, 8, 512]	0
Block 5	Conv2D	[14, 8, 512]	2359888
	Conv2D	[14, 8, 512]	2359808
	Conv2D	[14, 8, 512]	2359808
	MaxPooling 2D	[7, 4, 512]	0
FC Block	BatchNormalization	[7, 4, 512]	2048
	AveragePooling2D	[3, 2, 512]	0
	Flatten	[3072]	0
	Dense	[1024]	3146752
	Dropout	[1024]	0
	Dense	[2]	2050

VGG16 vs PedVis VGG16

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

	Layer	**Output Shape**	**parameters**
Block 1	Conv2D	[224, 128,64]	1792
	Conv2D	[224, 128, 64]	36928
	MaxPooling2D	[112, 64, 64]	0
Block 2	Conv2D	[112, 64, 128]	73856
	Conv2D	[112, 64, 128]	147584
	MaxPooling 2D	[56, 32, 128]	0
Block 3	Conv2D	[56, 32, 256]	295168
	Conv2D	[56,32,256]	590080
	Conv2D	[56,32,256]	590080
	MaxPooling 2D	[28,16,256]	0
Block 4	Conv2D	[28,16,512]	1188160
	Conv2D	[28,16, 512]	2359808
	Conv2D	[28 ,16, 512]	2359808
	MaxPooling 2D	[14, 8, 512]	0
Block 5	Conv2D	[14, 8, 512]	2359888
	Conv2D	[14, 8, 512]	2359808
	Conv2D	[14, 8, 512]	2359808
	MaxPooling 2D	[7, 4, 512]	0
FC Block	BatchNormalization	[7, 4, 512]	2048
	AveragePooling2D	[3, 2, 512]	0
	Flatten	[3072]	0
	Dense	[1024]	3146752
	Dropout	[1024]	0
	Dense	[2]	2050

Proposed solution (HyperParameters) : PedVis VGG16

Epochs : 100

Optimizer : Adam

Loss function : Categorical Cross-Entropy

Learning Rate : 0.0001

Image_size : 224 x 128

Batch_Size: 32

Proposed Solution with Limited Dataset

Train_images: 6000

Validation_images: 1200 (20%)

Test_images: 730

Performance Scores:

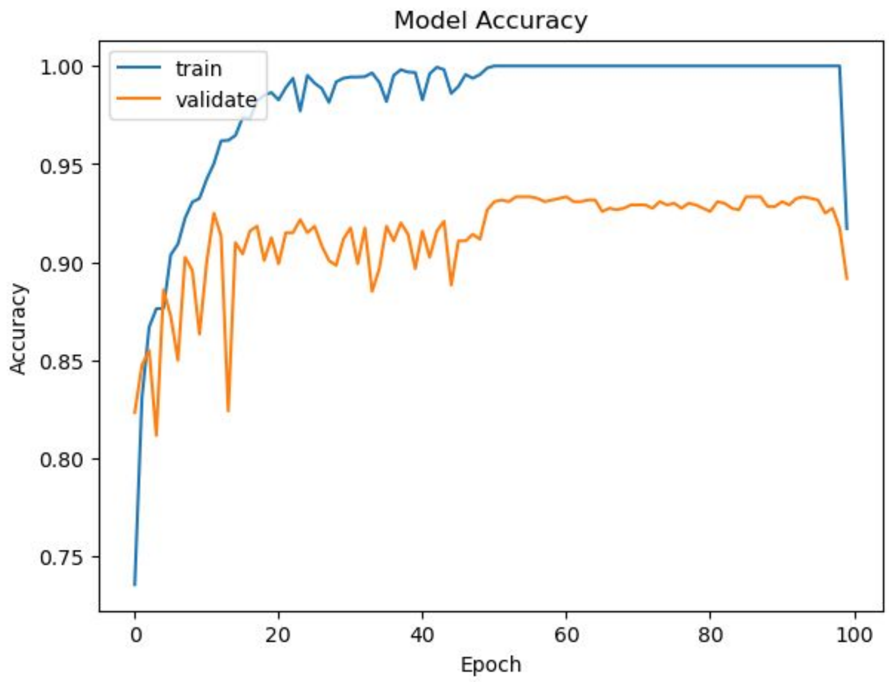
Accuracy: 0.9356164383561644

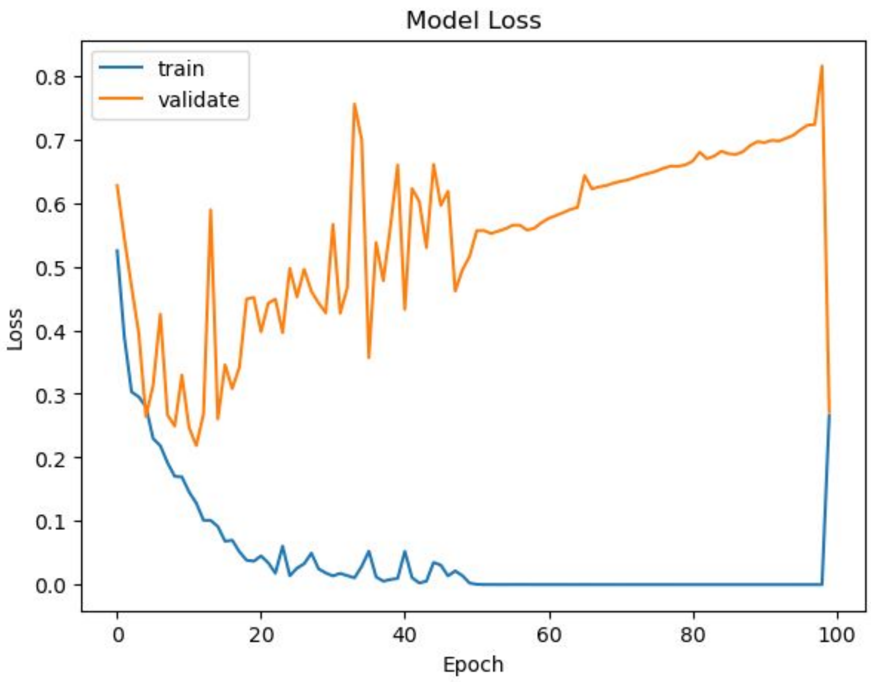
Precision: 0.8690095846645367

F1_score: 0.9204737732656515

Recall: 0.9784172661870504

Proposed Solution with Limited Dataset





Loss Function : Categorical Cross-Entropy

- Measures the difference between the predicted class probabilities and the true class probabilities.
- The intuition behind the categorical cross-entropy loss is to penalize the model for predicting a low probability for the correct class label.
- Rewards the model for predicting the correct class label with high probability and punishes the model for predicting the incorrect class label with low probability.

Formula for Categorical Cross-Entropy:

$$L(y, \hat{y}) = - \sum (y_i * \log(\hat{y}_i))$$

where **y** is a one-hot encoded vector representing the true class label and **ŷ** is the predicted probability distribution over all classes. The sum is taken over all classes.

Performance Score (Adam Optimizer)

Parameters:

Epochs: **50**

Optimizer : **Adam**

Loss function : **Categorical Cross-Entropy**

Learning Rate : **0.0001**

Image_size : **128 x 128**

train_images : **7120**

validation _images : **1780**

test_images: **1890**

Performance Score (Adam Optimizer)

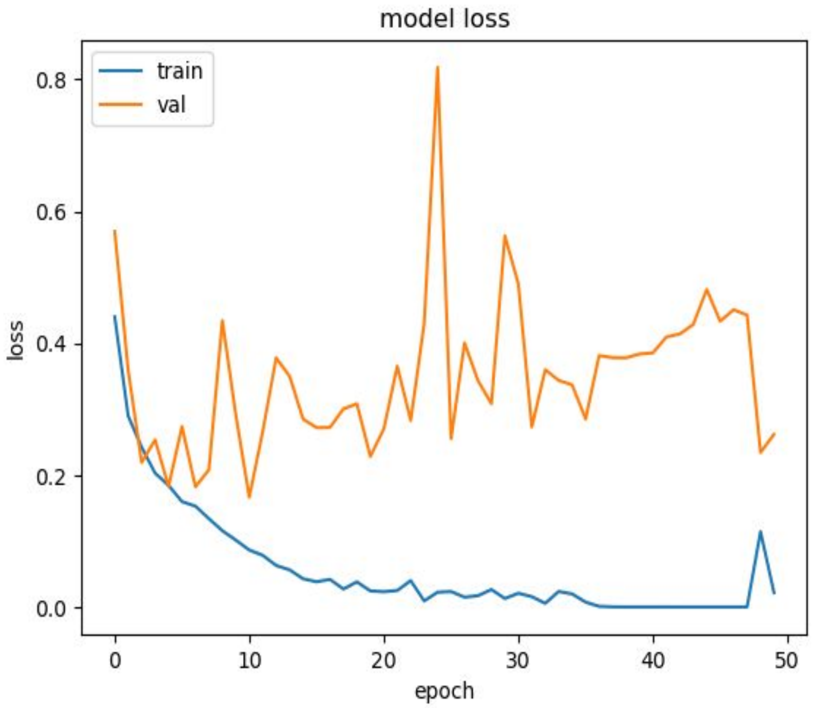
Accuracy : 0.8042328042328042

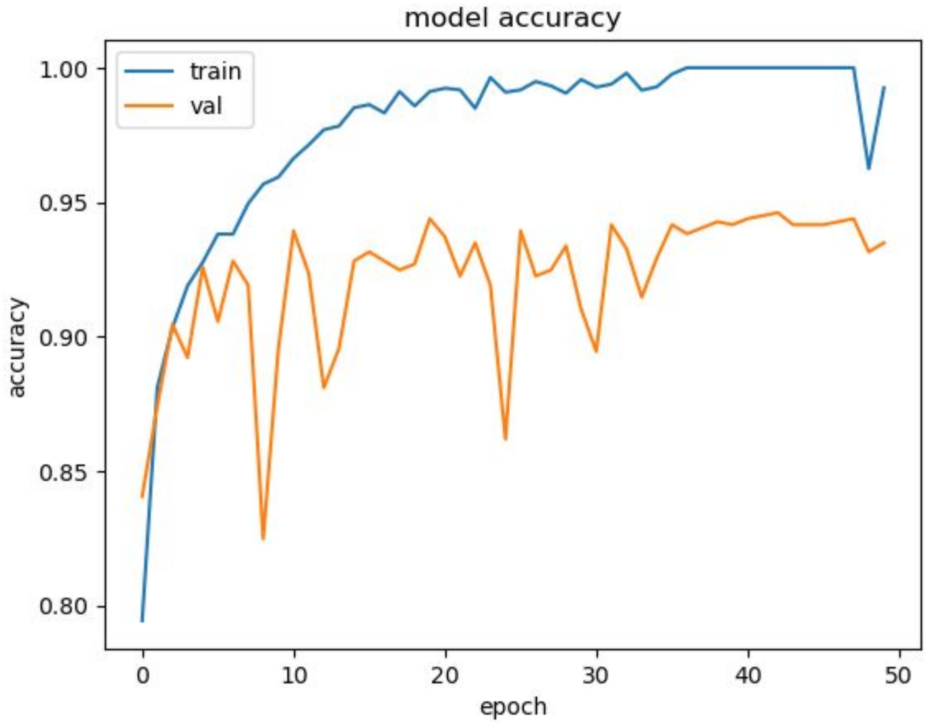
Recall : 0.80

F1_score : 0.79

Precision : 0.83

Training Score (Adam Optimizer)





Performance Score (SGD Optimizer)

Parameters:

Epochs: **50**

Optimizer : **SGD**

Loss function : **Categorical Cross-Entropy**

Learning Rate : **0.001**

Image_size : **128 x 128**

train_images : **7120**

validation _images : **1780**

test_images: **1890**

Performance Score (SGD Optimizer)

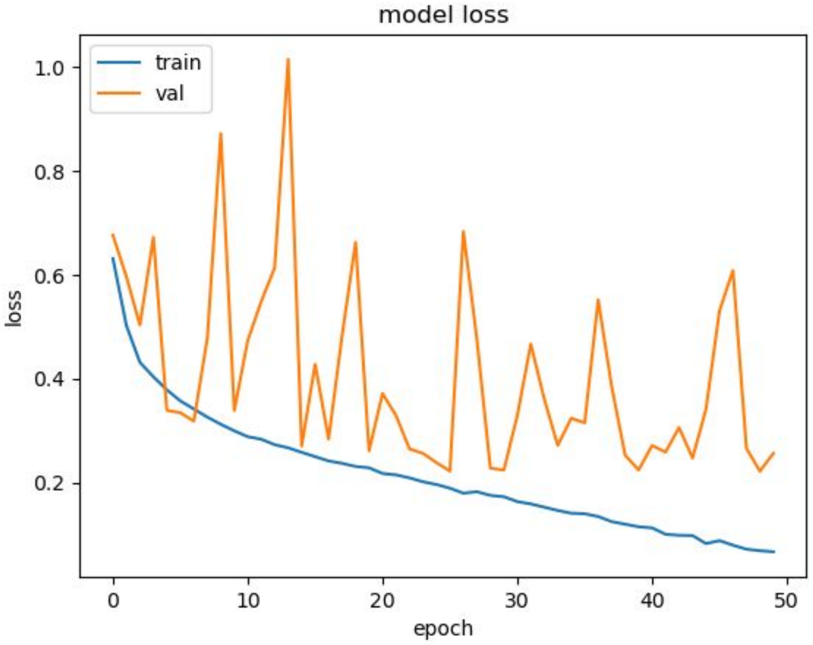
Accuracy: 0.7523809523809524

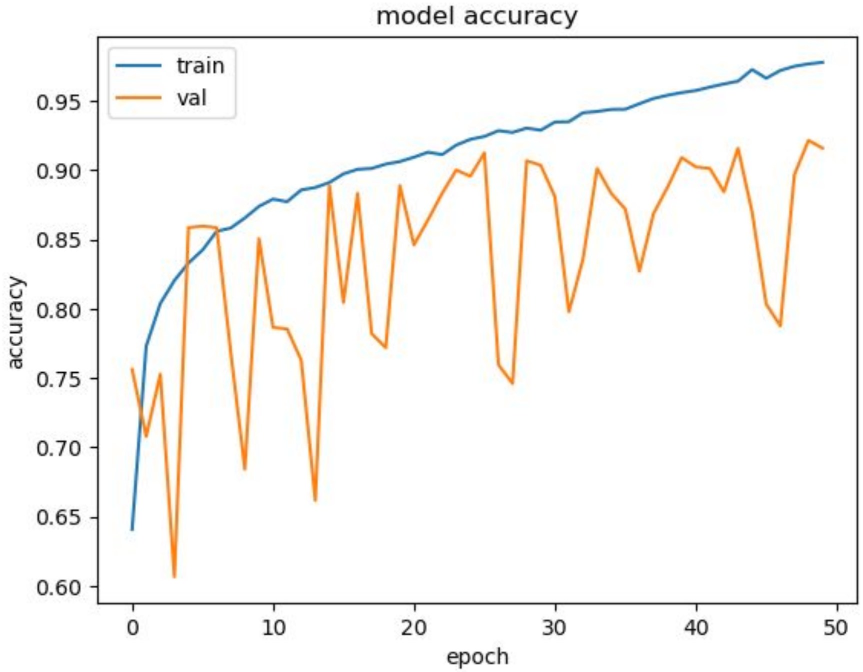
F1 Score: 0.6256

Recall: 0.4924433249370277

Precision: 0.8574561403508771

Training Score (SGD Optimizer)





Performance Score (SGD Optimizer)

Parameters:

Epochs: **50**

Optimizer : **SGD**

Loss function : **Categorical Cross-Entropy**

Learning Rate : **0.0001**

Image_size : **128 x 128**

train_images : **7120**

validation _images : **1780**

test_images: **1890**

Performance Score (SGD Optimizer)

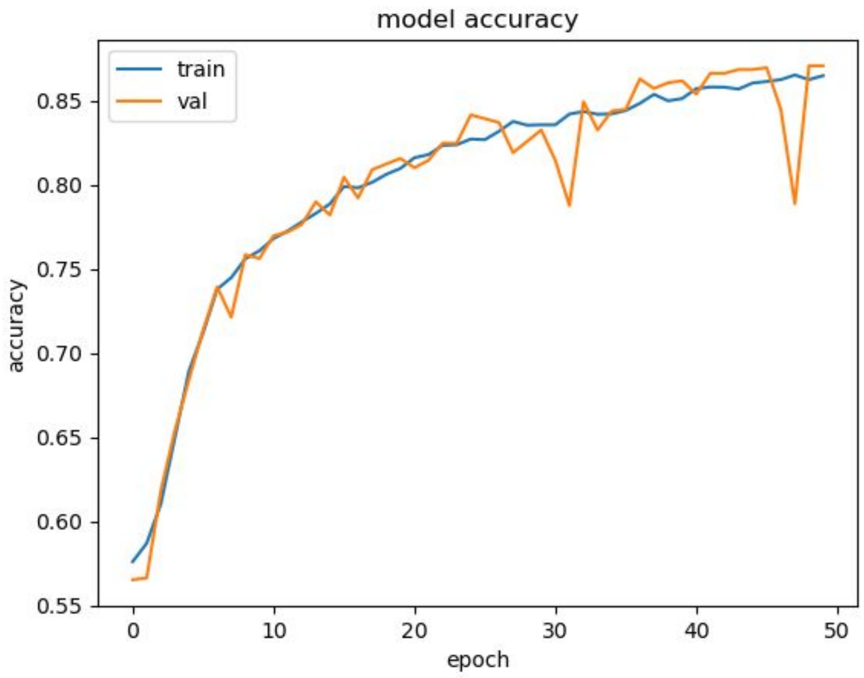
Accuracy: 0.6767195767195767

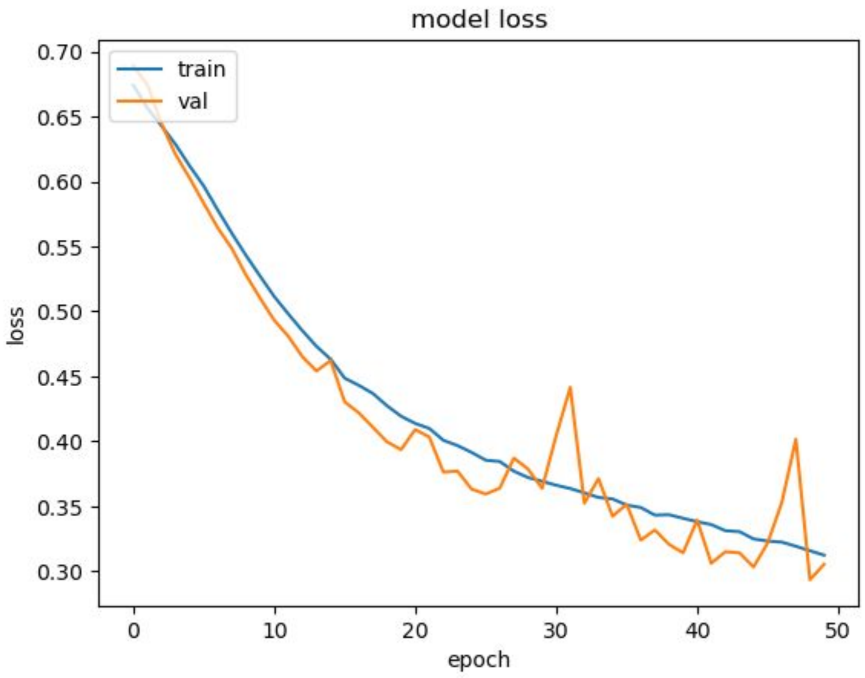
F1 Score: 0.47463456577816

Recall: 0.34760705289672544

Precision: 0.7479674796747967

Training Score(SGD Optimizer)





Resnet50(Architecture)

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix} \times 2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix} \times 3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix} \times 3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix} \times 3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix} \times 3$
conv3_x	28×28	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix} \times 2$	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix} \times 4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix} \times 4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix} \times 4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix} \times 8$
conv4_x	14×14	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix} \times 2$	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix} \times 6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix} \times 6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix} \times 23$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix} \times 36$
conv5_x	7×7	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix} \times 2$	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix} \times 3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix} \times 3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix} \times 3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix} \times 3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

Performance Score (Resnet50)

Parameters:

Epochs: **50**

Optimizer : **Adam**

Loss function : **Categorical Cross-Entropy**

Learning Rate : **0.001**

Image_size : **128 x 128**

train_images : **7120**

validation _images : **1780**

test_images: **1890**

Performance Score (Resnet50)

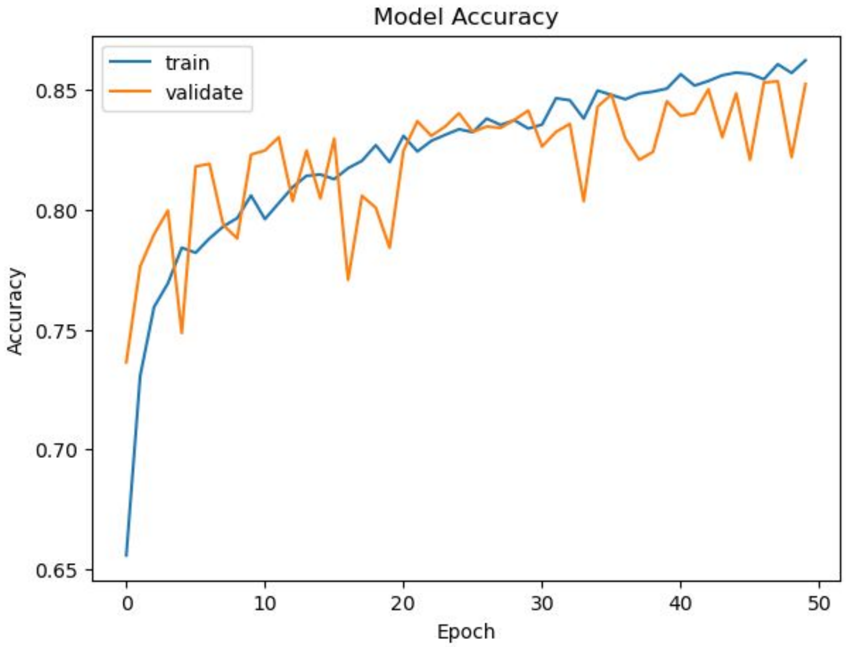
Accuracy:0.7074074074074074

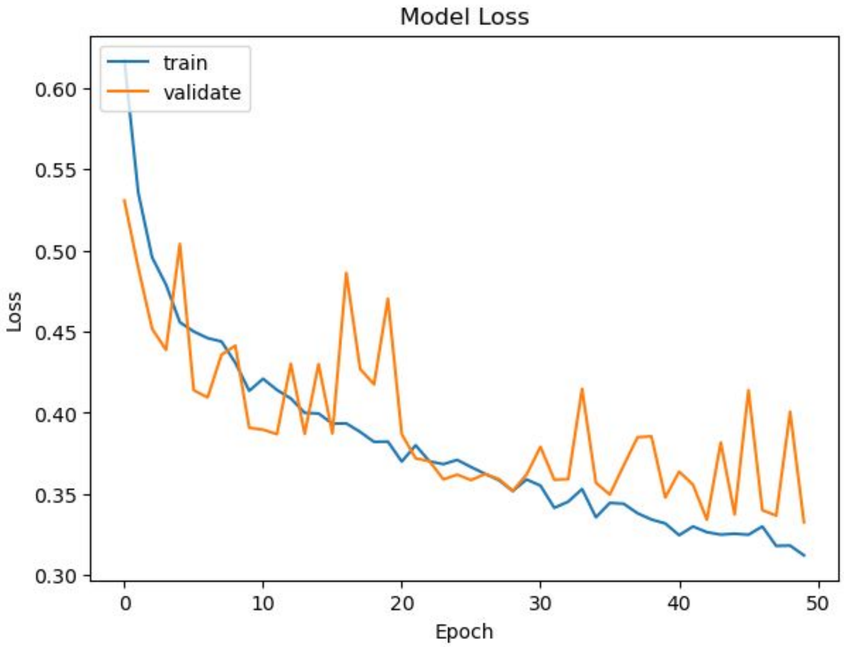
F1 Score: 0.5936811168258633

Recall: 0.5088161209068011

Precision: 0.7125220458553791

Training Score(Resnet50)





Performance Score (Incorporating VGG19)

Parameters:

Epochs: **50**

Optimizer : **Adam**

Loss function : **Categorical Cross-Entropy**

Learning Rate : **0.001**

Image_size : **128 x 128**

train_images : **7120**

validation _images : **1780**

test_images: **1890**

E 19 weight layers
conv3-64 conv3-64
conv3-128 conv3-128
conv3-256 conv3-256 conv3-256 **conv3-256**
conv3-512 conv3-512 conv3-512 **conv3-512**
conv3-512 conv3-512 conv3-512 **conv3-512**

	Layer	Output Shape	parameters
Block 1	Conv2D	[224, 128,64]	1792
	Conv2D	[224, 128, 64]	36928
	MaxPooling2D	[112, 64, 64]	0
Block 2	Conv2D	[112, 64, 128]	73856
	Conv2D	[112, 64, 128]	147584
	MaxPooling 2D	[56, 32, 128]	0
Block 3	Conv2D	[56, 32, 256]	295168
	Conv2D	[56,32,256]	590080
	Conv2D	[56,32,256]	590080
	MaxPooling 2D	[28,16,256]	0
Block 4	Conv2D	[28,16,512]	1188160
	Conv2D	[28,16, 512]	2359808
	Conv2D	[28 ,16, 512]	2359808
	MaxPooling 2D	[14, 8, 512]	0
Block 5	Conv2D	[14, 8, 512]	2359888
	Conv2D	[14, 8, 512]	2359808
	Conv2D	[14, 8, 512]	2359808
	MaxPooling 2D	[7, 4, 512]	0
FC Block	BatchNormalization	[7, 4, 512]	2048
	AveragePooling2D	[3, 2, 512]	0
	Flatten	[3072]	0
	Dense	[1024]	3146752
	Dropout	[1024]	0
	Dense	[2]	2050

Performance Score (Incorporating VGG19)

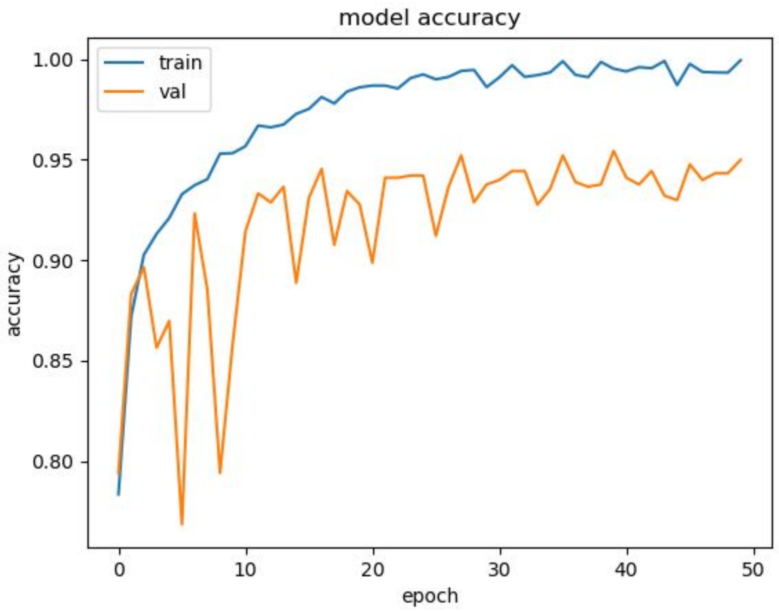
Accuracy: 0.8396825396825397

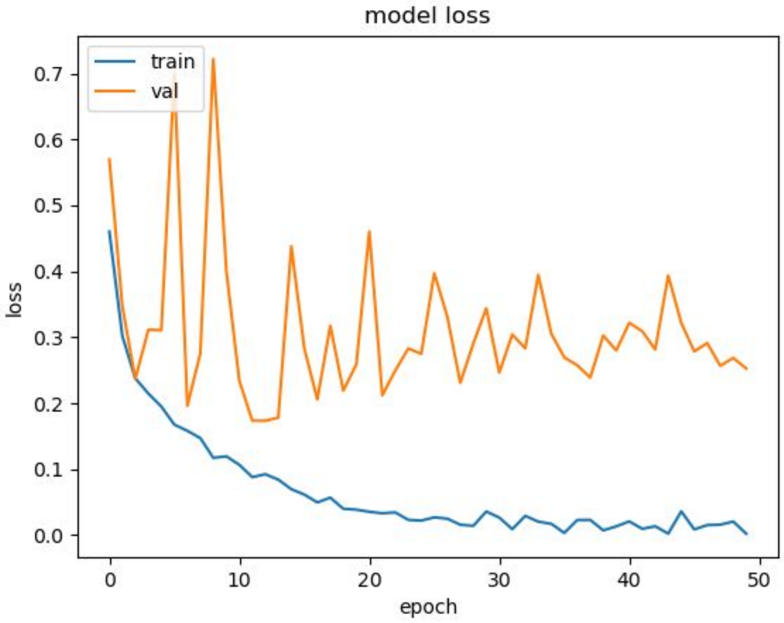
F1 Score: 0.84

Recall: 0.82

Precision: 0.85

Training Score (Incorporating VGG19)





Performance Score (One extra block of convolution)

Parameters:

Epochs: **50**

Optimizer : **Adam**

Loss function : **Categorical Cross-Entropy**

Learning Rate : **0.001**

Image_size : **128 x 128**

train_images : **7120**

validation _images : **1780**

test_images: **1890**

Conv2D
Conv2D
Conv2D

	Layer	Output Shape	parameters
Block 1	Conv2D	[224, 128,64]	1792
	Conv2D	[224, 128, 64]	36928
	MaxPooling2D	[112, 64, 64]	0
Block 2	Conv2D	[112, 64, 128]	73856
	Conv2D	[112, 64, 128]	147584
	MaxPooling 2D	[56, 32, 128]	0
Block 3	Conv2D	[56, 32, 256]	295168
	Conv2D	[56,32,256]	590080
	Conv2D	[56,32,256]	590080
	MaxPooling 2D	[28,16,256]	0
Block 4	Conv2D	[28,16,512]	1188160
	Conv2D	[28,16, 512]	2359808
	Conv2D	[28 ,16, 512]	2359808
	MaxPooling 2D	[14, 8, 512]	0
Block 5	Conv2D	[14, 8, 512]	2359888
	Conv2D	[14, 8, 512]	2359808
	Conv2D	[14, 8, 512]	2359808
	MaxPooling 2D	[7, 4, 512]	0
FC Block	BatchNormalization	[7, 4, 512]	2048
	AveragePooling2D	[3, 2, 512]	0
	Flatten	[3072]	0
	Dense	[1024]	3146752
	Dropout	[1024]	0
	Dense	[2]	2050

Performance Score (One extra block of convolution)

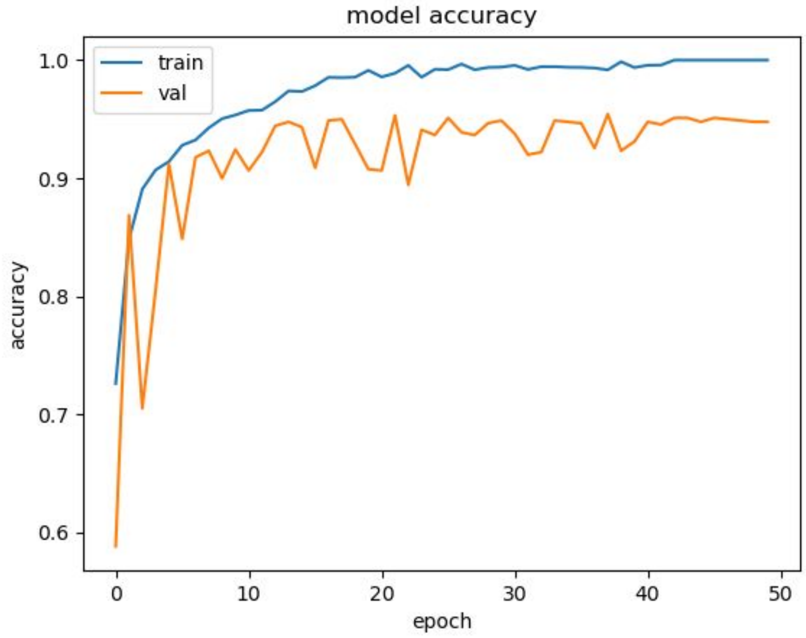
Accuracy: 0.828042328042328

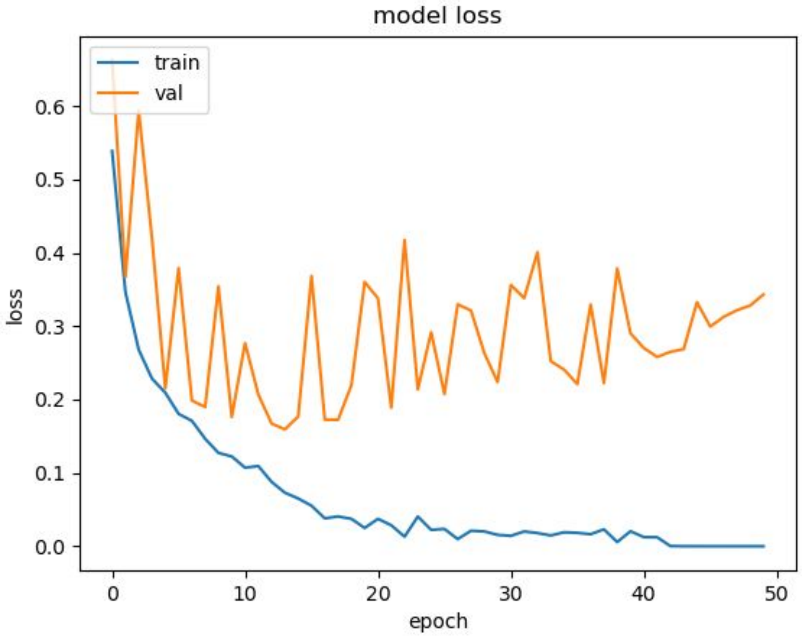
F1 Score: 0.82

Recall: 0.83

Precision: 0.84

Training Score (One extra block of Convolution)





Comparison

1.Adding one block to the PedVis architecture gives better values than the proposed PedVis Architecture with 9000 training dataset.

2.Learning rate = 0.0001 performs better than learning rate = 0.001.

3.Adam optimizer works better than SGD optimizer.

4.Image_size = (128,128) is given for accommodating more train images instead of image_size = (224,128) which gives less accuracy.

5.VGG-19 gives the best accuracy among PedVis and one-block added architecture.

Comparison with State-of-the-art methods

We used **Resnet-50.**

Performance metrics of Resnet-50 do not have a higher score than the proposed architecture for this dataset.

	PedVis	Resnet-50
Precision	0.8690095846645367	0.7125220458553791
F1_score	0.9204737732656515	0.5936811168258633
Recall	0.9784172661870504	0.5088161209068011
Accuracy	0.9356164383561644	0.7074074074074074

Challenges

- **Initial dataset was too small with 632 images to train, all having positive labels. So we had to combine several datasets.**

- **All the models were run in Kaggle and we faced some memory issues which caused us to train on 6000 images, instead of 9000. Eventually, we solved this issue by reducing the input image size from (224 x 128) to (128 x 128)**