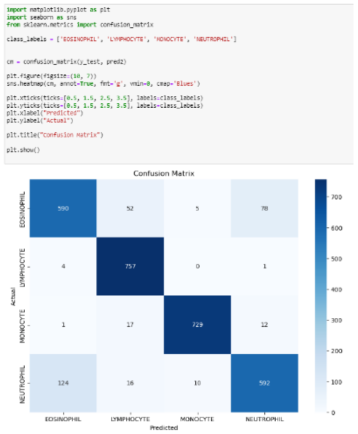
Project Development Phase
Model Performance Test

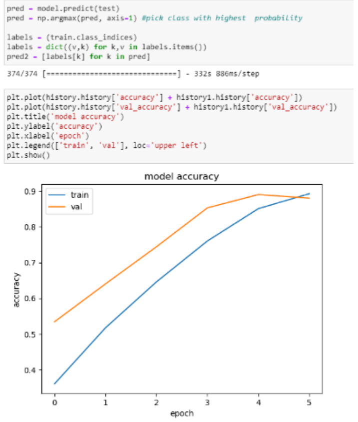
Date	10 January 2026
Team ID	LTVIP2025TMIDS66117
Project Name	Hematovision: advanced blood cell classification using transfer learning
Maximum Marks	10 Marks

Model Performance Testing:

Project team shall fill the following information in model performance testing template.

S.No.	Parameter	Values	Screenshot

1.	Metrics	**Classification Model:**
		Accuray Score- 0.893
			
			`import matplotlib.pyplot as plt`
			`import seaborn as sns`
			`from sklearn.metrics import confusion_matrix`
			
			`class_labels = ['EOSINOPHIL', 'LYMPHOCYTE', 'MONOCYTE', 'NEUTROPHIL']`
			
			`cm = confusion_matrix(y_test, pred2)`
			
			`plt.figure(figsize=(10, 7))`
			`sns.heatmap(cm, annot=True, fmt='g', vmin=0, cmap='Blues')`
			
			`plt.xticks(ticks=[0.5, 1.5, 2.5, 3.5], labels=class_labels)`
			`plt.yticks(ticks=[0.5, 1.5, 2.5, 3.5], labels=class_labels)`
			`plt.xlabel("Predicted")`
			`plt.ylabel("Actual")`
			
			`plt.title("Confusion Matrix")`
			
			`plt.show()`
			 Confusion Matrix
			EOSINOPHIL 590 52 5 78
			LYMPHOCYTE 4 757 0 1
			MONOCYTE 1 17 729 12
			NEUTROPHIL 124 16 10 592
			EOSINOPHIL LYMPHOCYTE MONOCYTE NEUTROPHIL
			Predicted
			Actual
			700 600 500 400 300 200 100 0

2.	Tune the Model	Hyperparameter Tuning - The notebook primarily focuses on training the added dense layers with a pre-trained MobileNetV2 model (frozen base layers). Adam optimizer was used with categorical crossentropy loss. The training ran for 15 epochs, with the best validation accuracy observed around epoch 11.<br>Validation Method - A validation split of 0.2 was used during image data generation (validation_split=0.2).	
			`pred = model.predict(test)`
			`pred = np.argmax(pred, axis=1) #pick class with highest  probability`
			
			`labels = (train.class_indices)`
			`labels = dict((v,k) for k,v in labels.items())`
			`pred2 = [labels[k] for k in pred]`
			
			`374/374 [==============================] - 332s 886ms/step`
			
			`plt.plot(history.history['accuracy'] + history1.history['accuracy'])`
			`plt.plot(history.history['val_accuracy'] + history1.history['val_accuracy'])`
			`plt.title('model accuracy')`
			`plt.ylabel('accuracy')`
			`plt.xlabel('epoch')`
			`plt.legend(['train', 'val'], loc='upper left')`
			`plt.show()`
			 model accuracy
			train
			val
			accuracy
			0.9 0.8 0.7 0.6 0.5 0.4
			0 1 2 3 4 5
			epoch