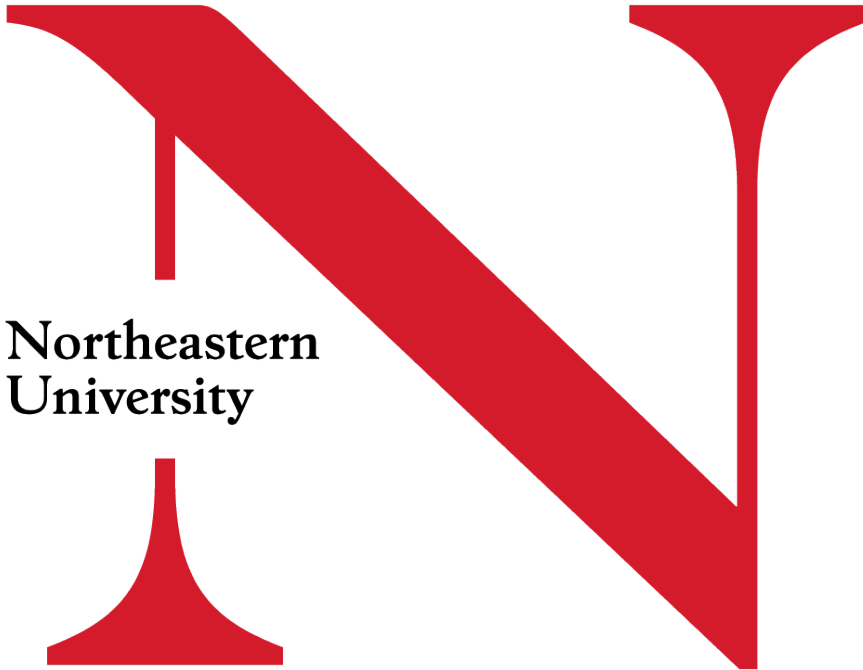
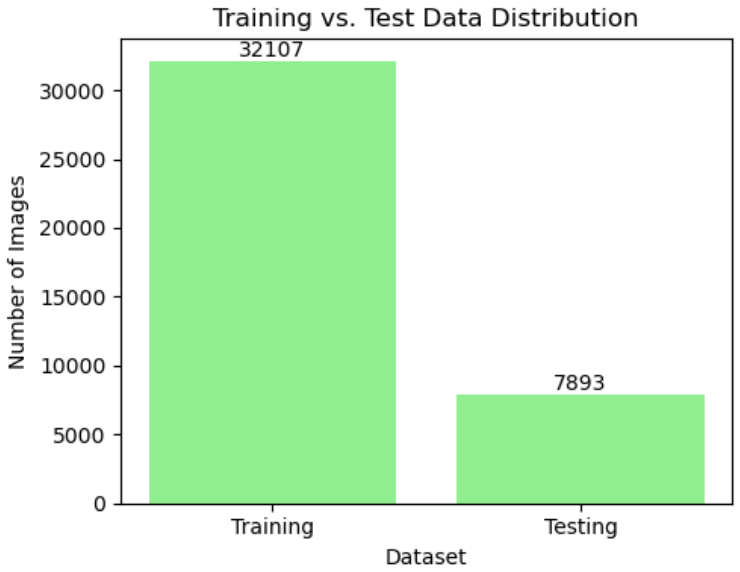
Concrete Surface Crack Detection



1. Data Inspection

Input Size: 227x227
Color Mode: RGB
No Corrupted Images



Sample Positive Training Images

0.jpg

1.jpg

10.jpg

100.jpg

1000.jpg

Sample Negative Training Images

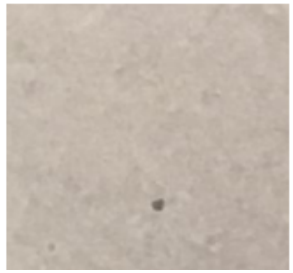
0.jpg

1.jpg

10.jpg

100.jpg

1000.jpg



Sample Test Images

1000016480768558.png

1000094138882661.png

100010347143101.png

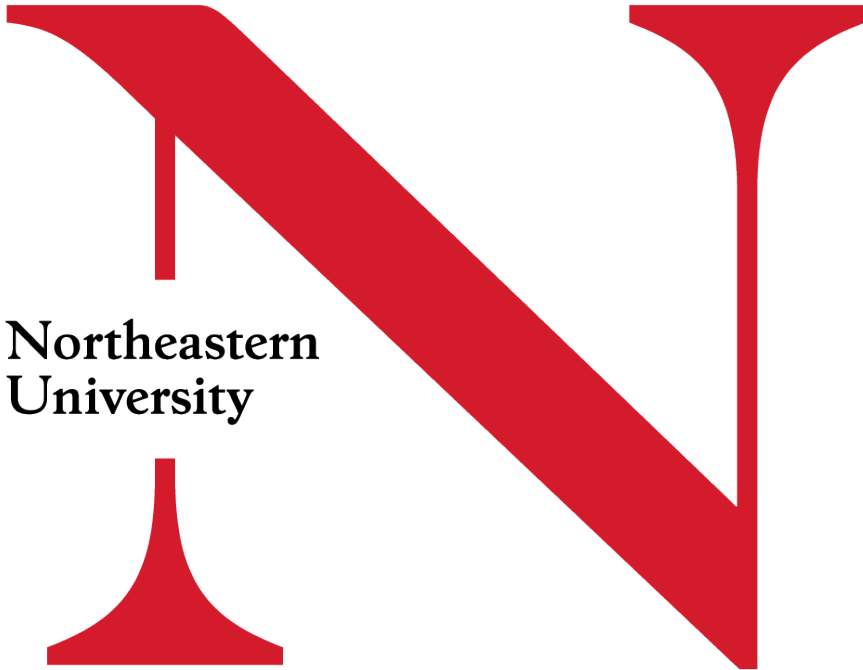
1000228345963509.png

10002760868742.png

2. Data Preprocessing

- ImageDataGenerator
- Normalize
- Flow_from_directory: train and validation data
- Flow_from_dataframe: test data
- No image augmentation

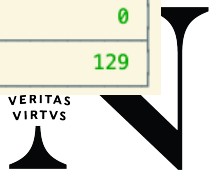
3. Custom Model

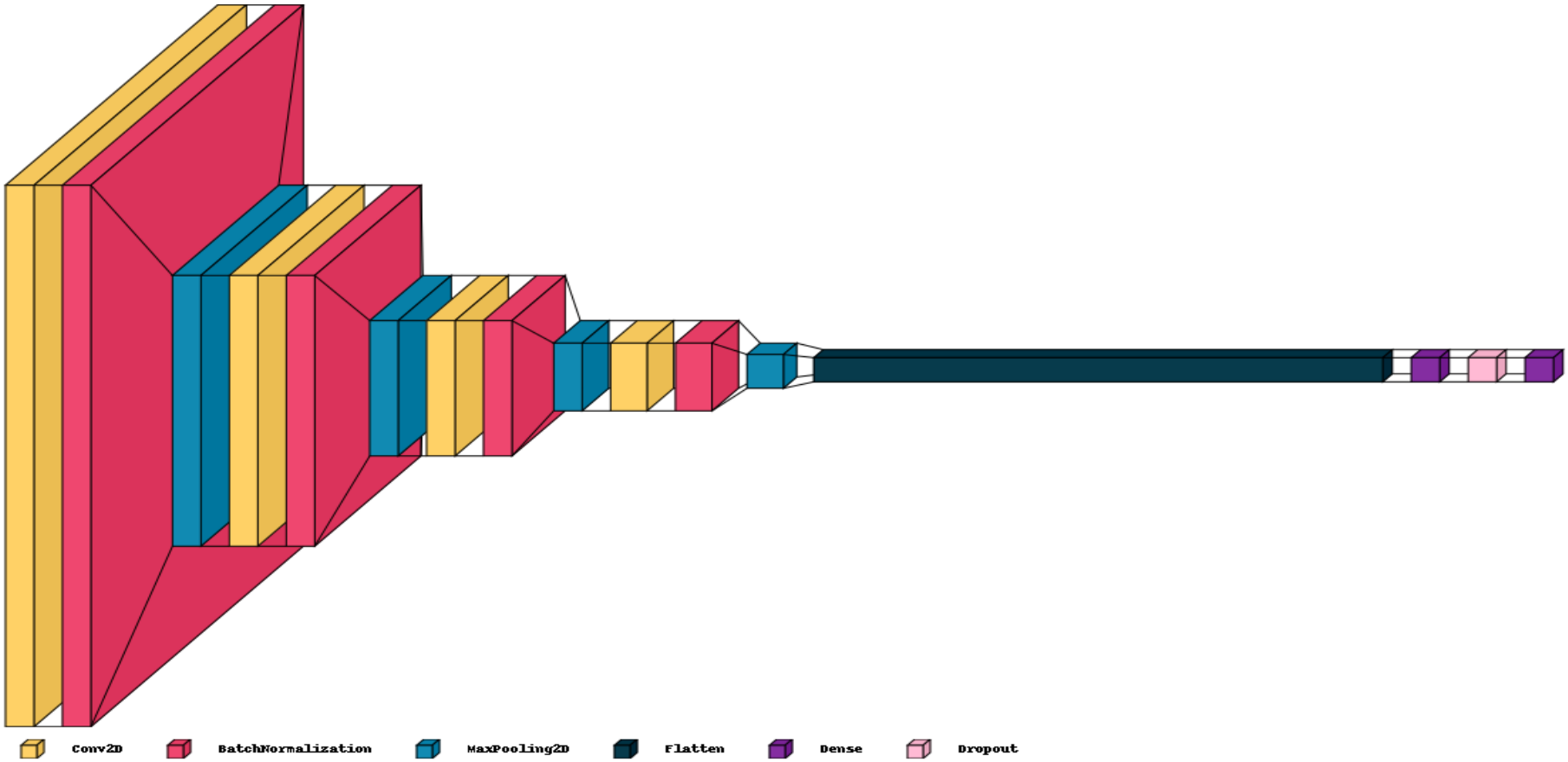


3.1 Model Architecture

- **Input Layer**
  128x128.
- **Convolutional Layer 1：** 32 x (3, 3), ReLU,
  padding = same.
  Batch Normalization.
  MaxPooling (2, 2).
- **Convolutional Layer 2：** 64 x (3, 3), ReLU,
  padding = same.
  Batch Normalization.
  MaxPooling (2, 2).
- **Convolutional Layer 3：** 128 x (3, 3), ReLU,
  padding = same.
  Batch Normalization.
  MaxPooling (2, 2).
- **Convolutional Layer 4：** 256 x (3, 3), ReLU,
  padding = same.
  Batch Normalization.
  MaxPooling (2, 2).
- **Fully Connected Layers**
  Flatten.
  Dense(128), ReLU.
  Dropout(50%).
- **Output Layer**
  Dense(1), sigmoid.

Layer (type)	Output Shape	Param #
conv2d_4 (Conv2D)	(None, 128, 128, 32)	896
batch_normalization_10 (BatchNormalization)	(None, 128, 128, 32)	128
max_pooling2d_4 (MaxPooling2D)	(None, 64, 64, 32)	0
conv2d_5 (Conv2D)	(None, 64, 64, 64)	18,496
batch_normalization_11 (BatchNormalization)	(None, 64, 64, 64)	256
max_pooling2d_5 (MaxPooling2D)	(None, 32, 32, 64)	0
conv2d_6 (Conv2D)	(None, 32, 32, 128)	73,856
batch_normalization_12 (BatchNormalization)	(None, 32, 32, 128)	512
max_pooling2d_6 (MaxPooling2D)	(None, 16, 16, 128)	0
conv2d_7 (Conv2D)	(None, 16, 16, 256)	295,168
batch_normalization_13 (BatchNormalization)	(None, 16, 16, 256)	1,024
max_pooling2d_7 (MaxPooling2D)	(None, 8, 8, 256)	0
flatten_1 (Flatten)	(None, 16384)	0
dense_14 (Dense)	(None, 128)	2,097,280
dropout_7 (Dropout)	(None, 128)	0
dense_15 (Dense)	(None, 1)	129





LVX
VERITAS
VIRTVS

3.2 Model Training

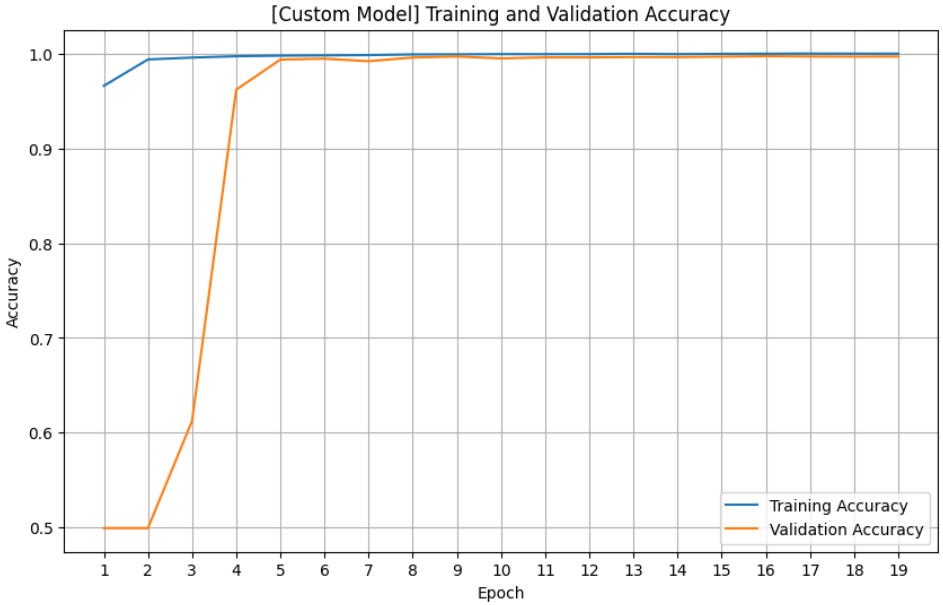
Set Hyperparameters
IMAGE_SIZE = (128, 128)
BATCH_SIZE = 256
EPOCHS = 30

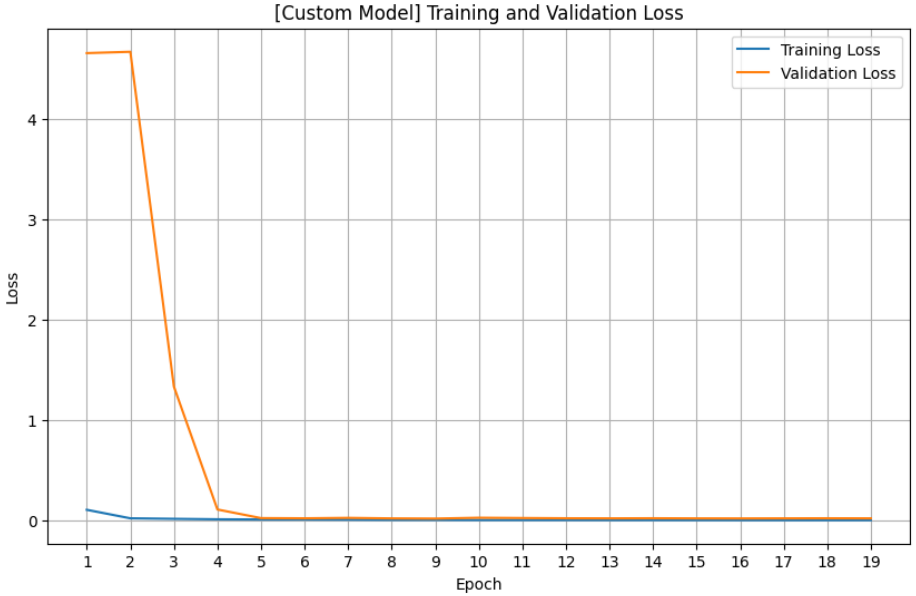
EarlyStopping
monitor='val_loss'
patience=10,

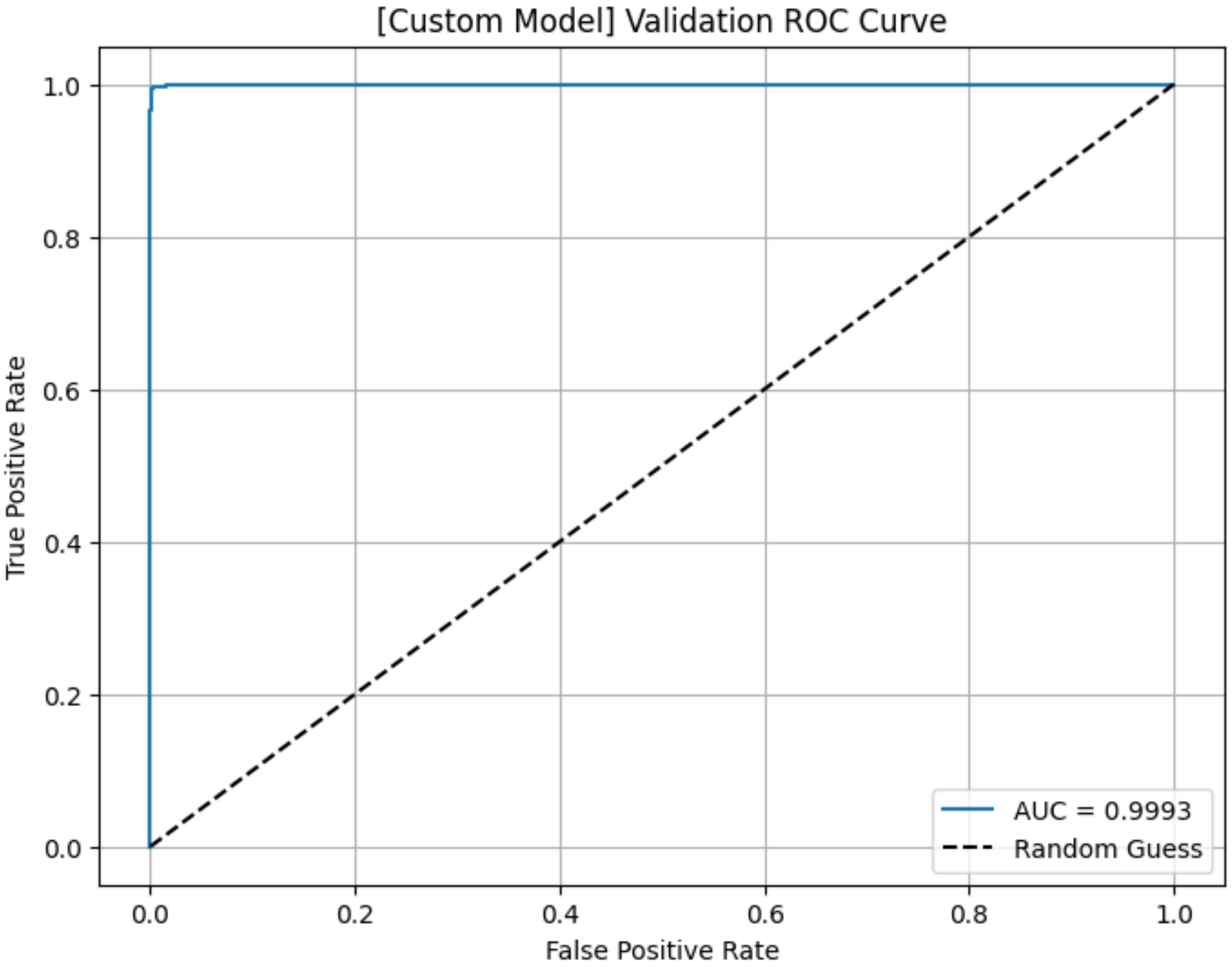
Model Compile
optimizer=Adam(learning_rate=1e-4),
loss='binary_crossentropy',
metrics=['accuracy']

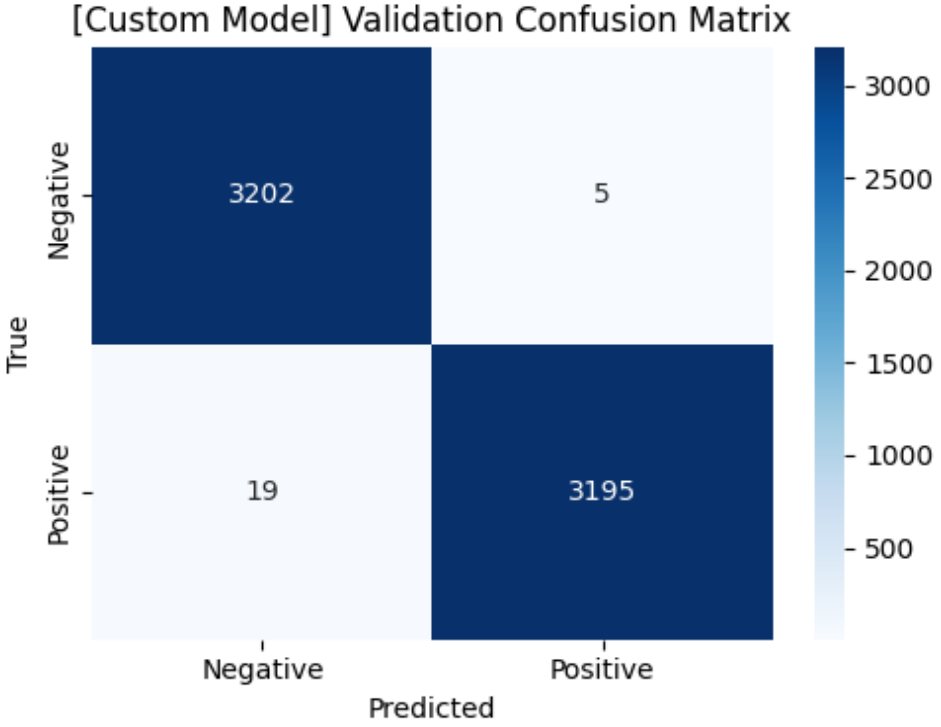
ReduceLROnPlateau
monitor='val_loss',
factor=0.2,
patience=5,
cooldown=3,
min_lr=1e-7

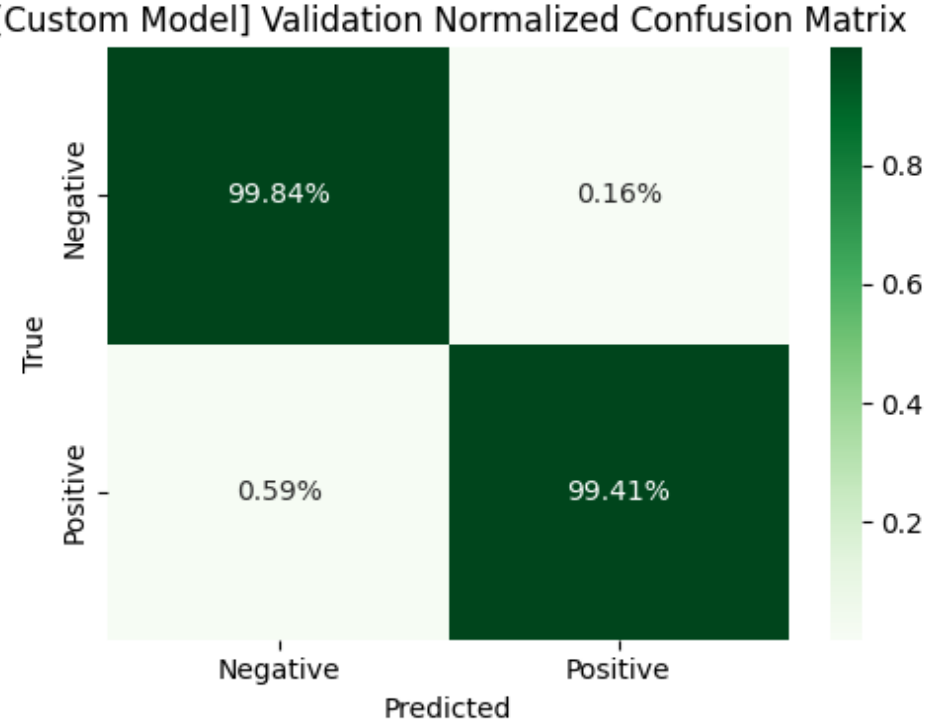
3.3 Model Evaluation






[Custom Model] Validation ROC Curve
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
AUC = 0.9993
Random Guess


[Custom Model] Validation Confusion Matrix
Negative
Positive
True
3202
5
19
3195
Negative
Positive
Predicted
3000
2500
2000
1500
1000
500


[Custom Model] Validation Normalized Confusion Matrix
Negative
Positive
True
99.84%
0.16%
0.59%
99.41%
Negative
Positive
Predicted
0.8
0.6
0.4
0.2

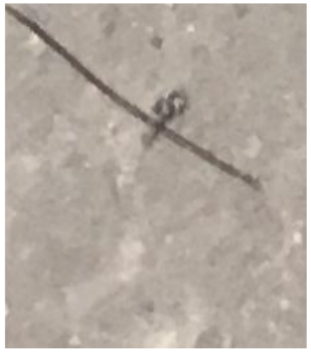
False Positives (FP)

True: 0, Pred: 1
Prob: 0.56

True: 0, Pred: 1
Prob: 0.70

True: 0, Pred: 1
Prob: 0.91

True: 0, Pred: 1
Prob: 1.00



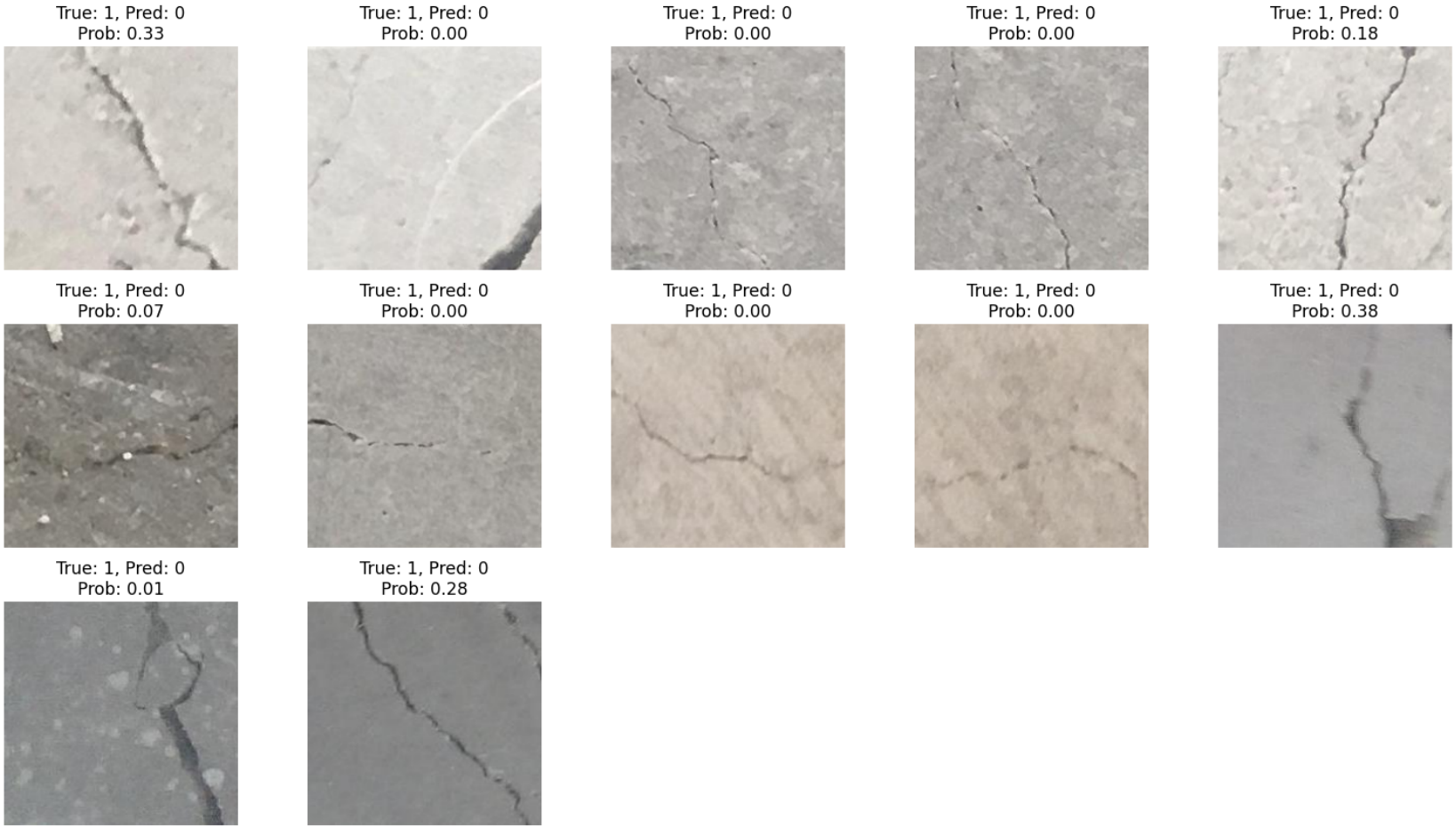
True: 0, Pred: 1
Prob: 0.98

True: 0, Pred: 1
Prob: 0.70

True: 0, Pred: 1
Prob: 1.00

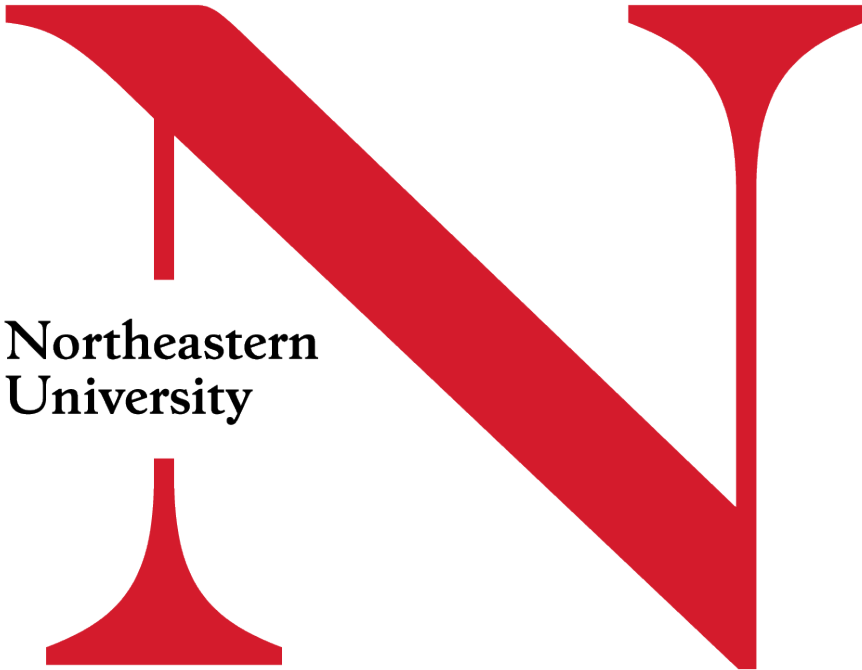
True: 0, Pred: 1
Prob: 0.61

False Negatives (FN)

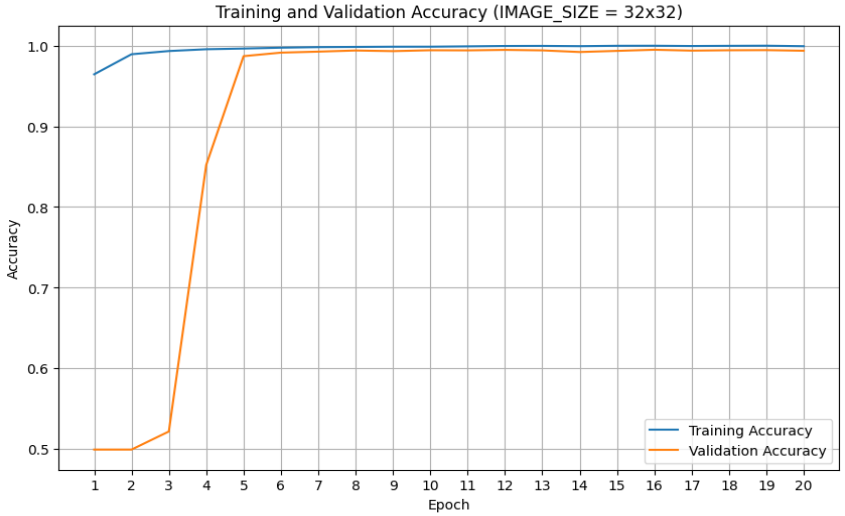


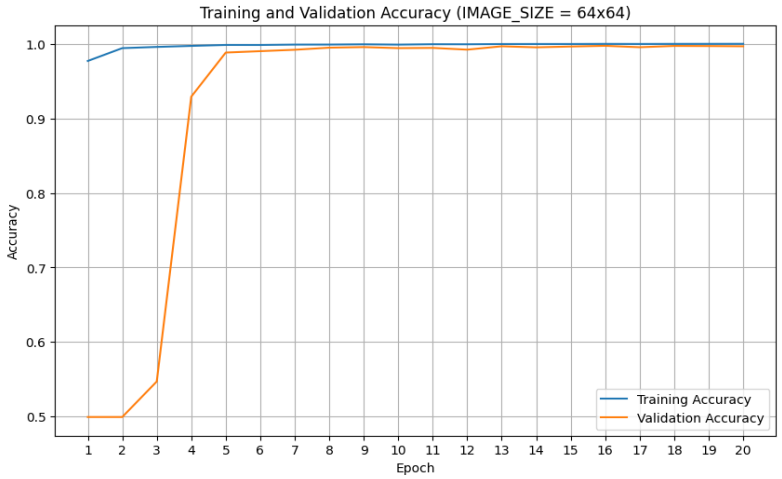
LVX
VERITAS
VIRTVS

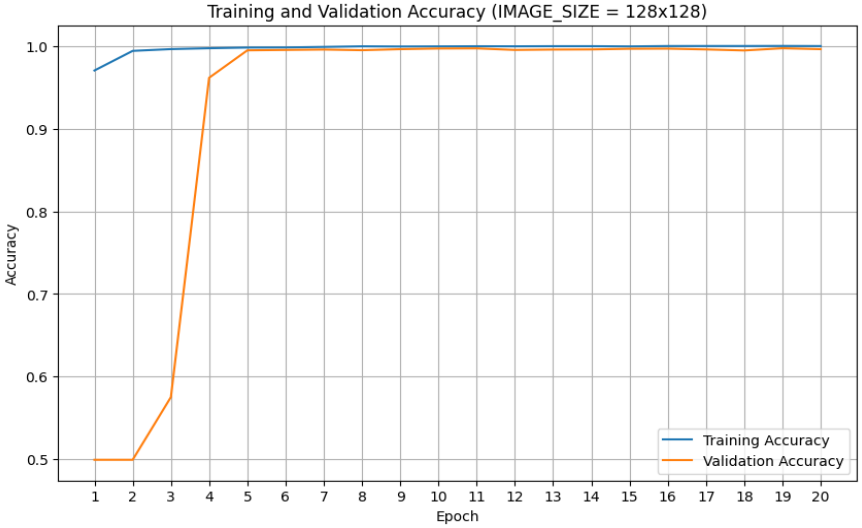
4.Test

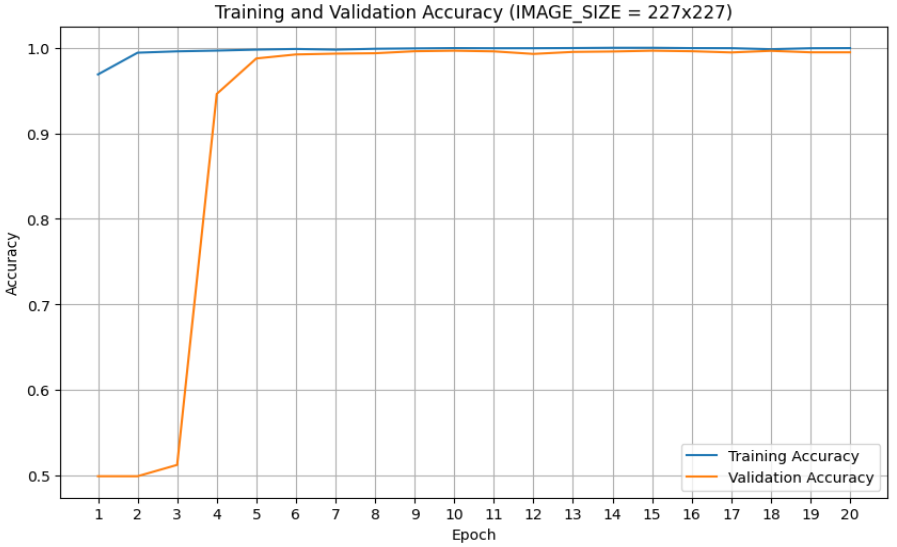


4.1 Image Size

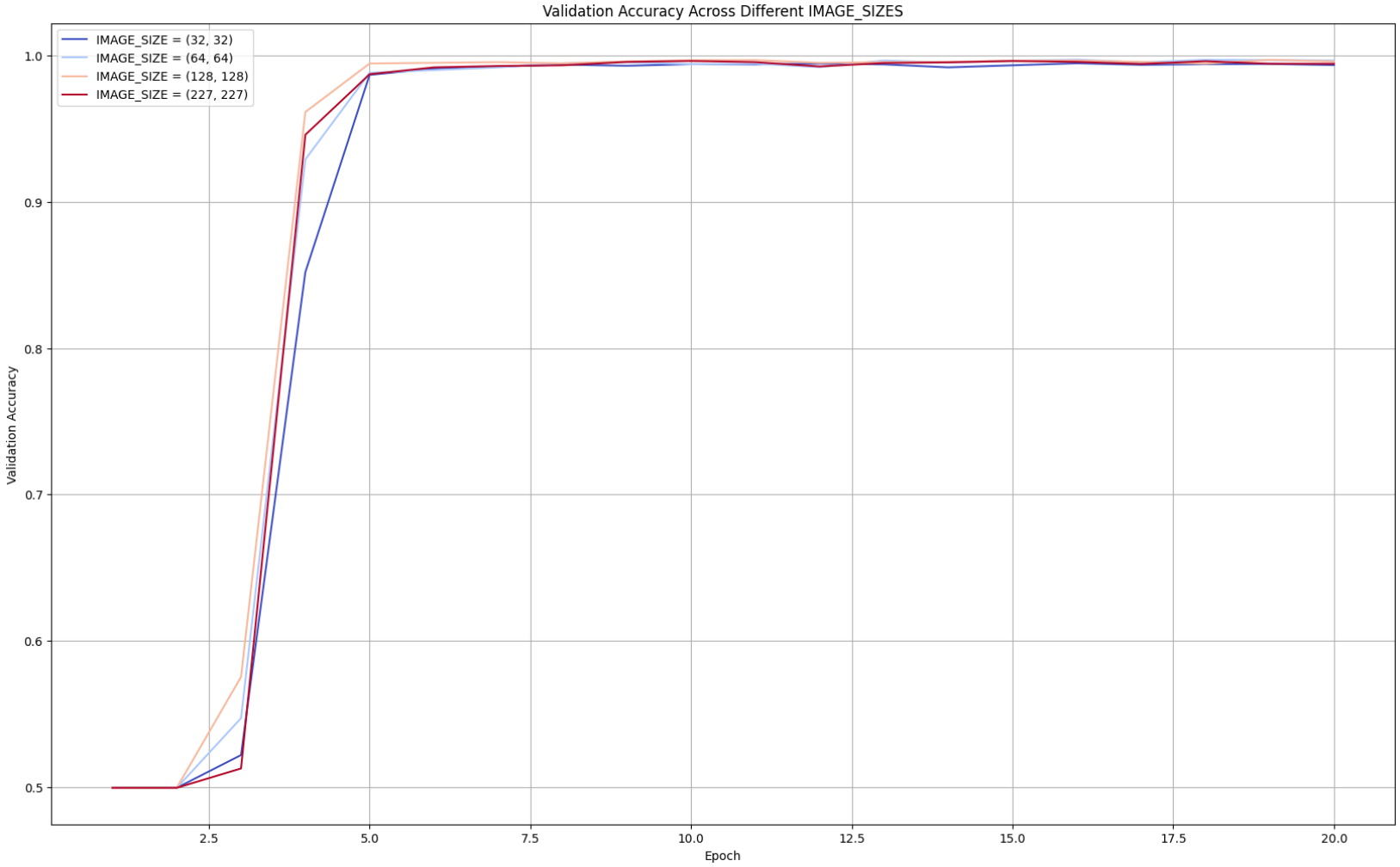






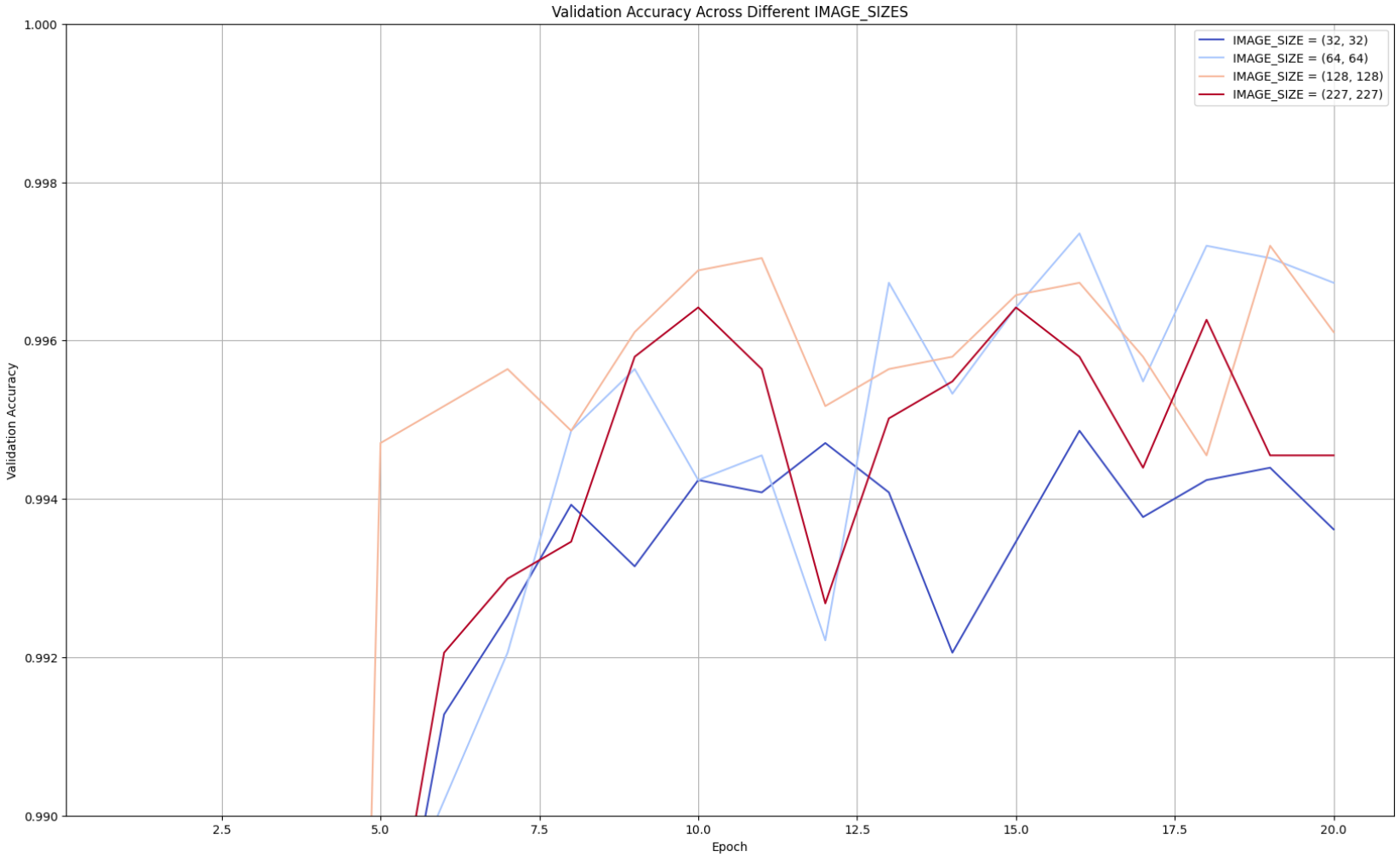


batch_size = 128
lr=1e-4
epoc=20



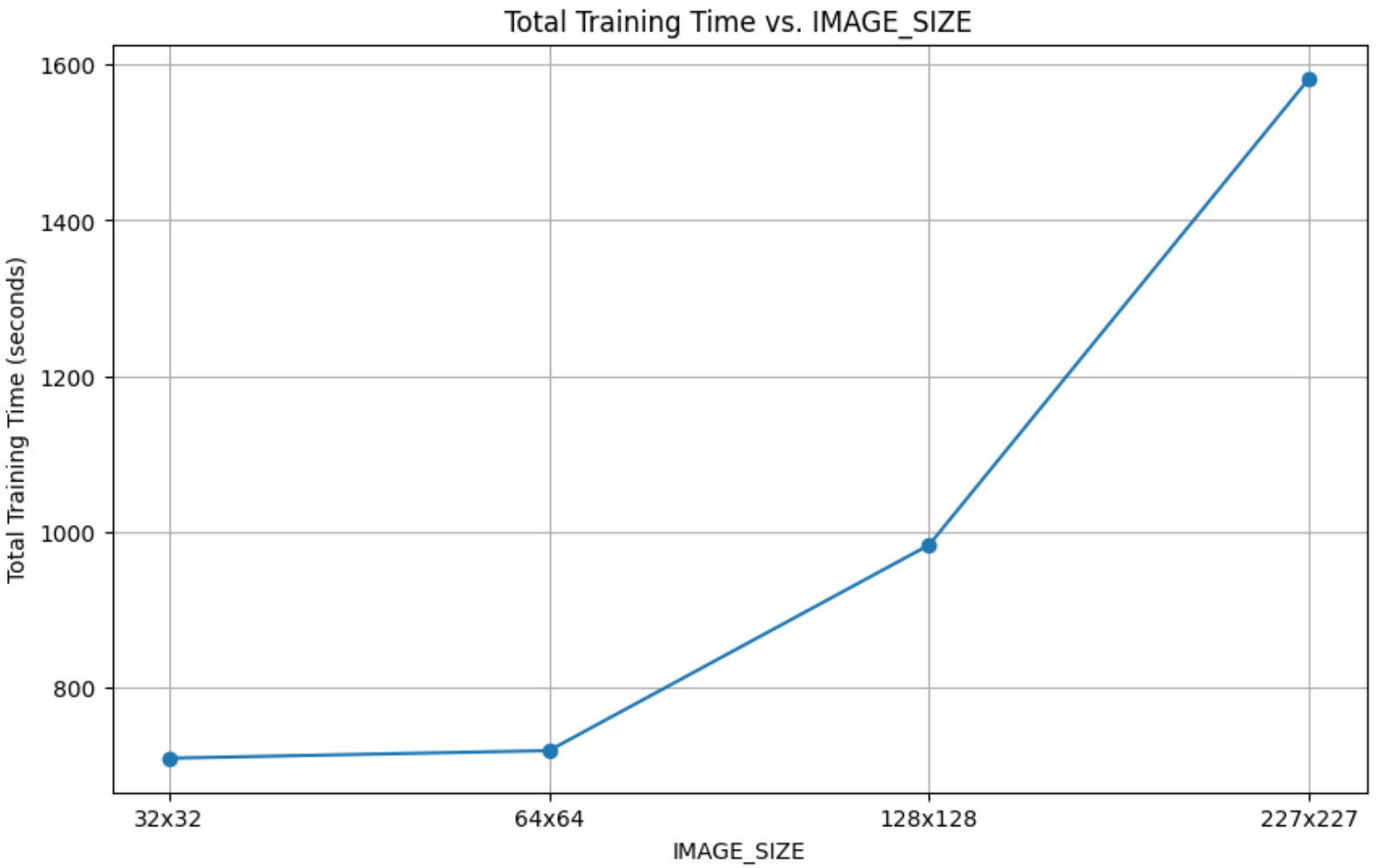
LVX
VERITAS
VIRTVS

batch_size = 128
lr=1e-4
epoc=20



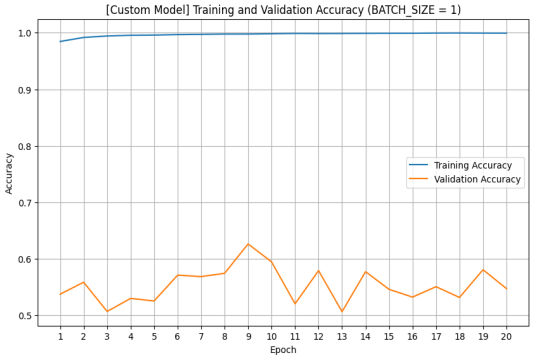
LVX
VERITAS
VIRTVS

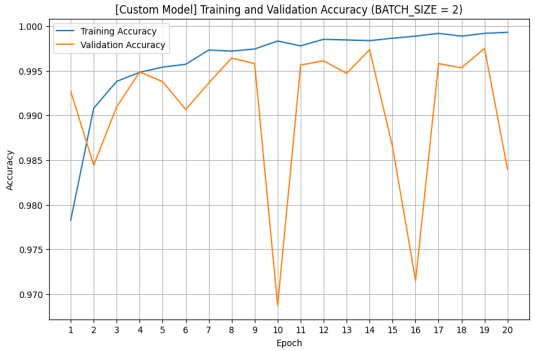
batch_size = 128
lr=1e-4
epoc=20

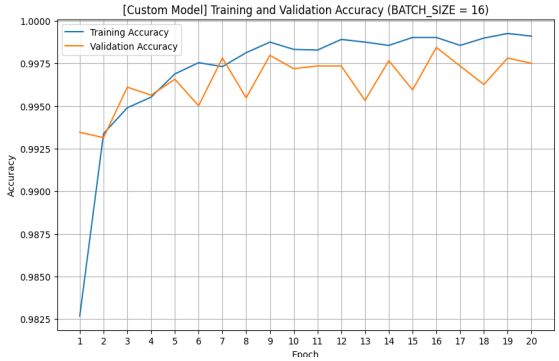


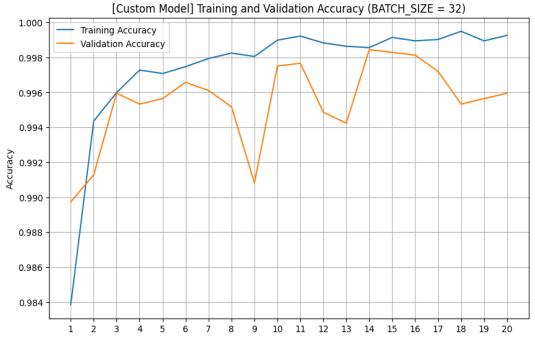
4.2 Batch Size

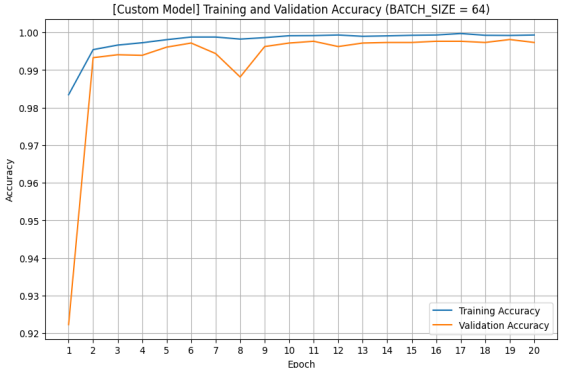
input=128x128
lr=1e-4
epoc=20

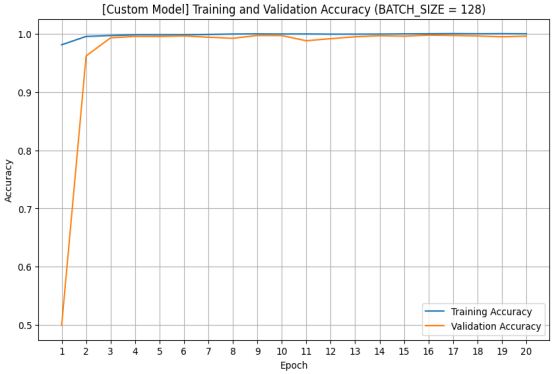


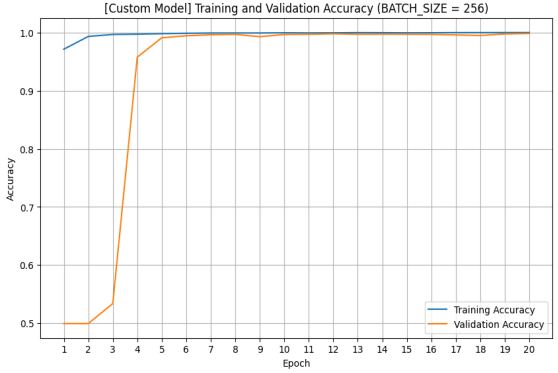


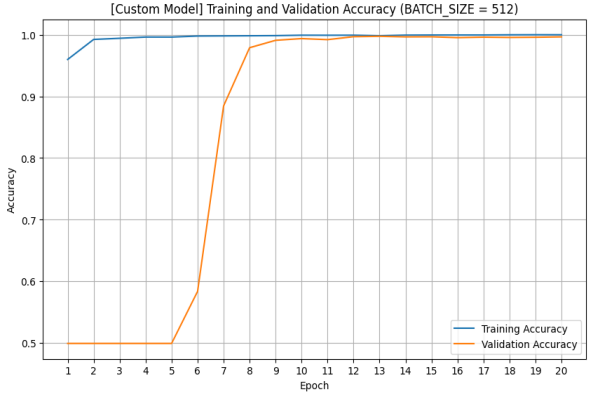


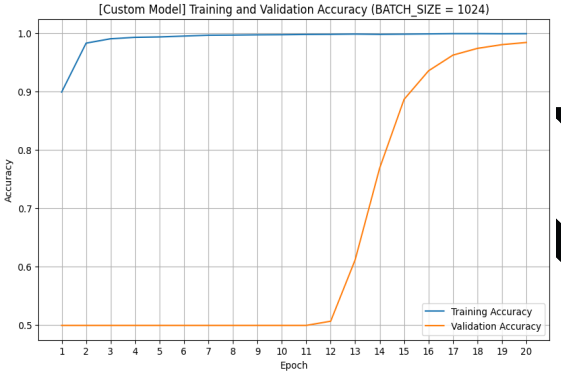




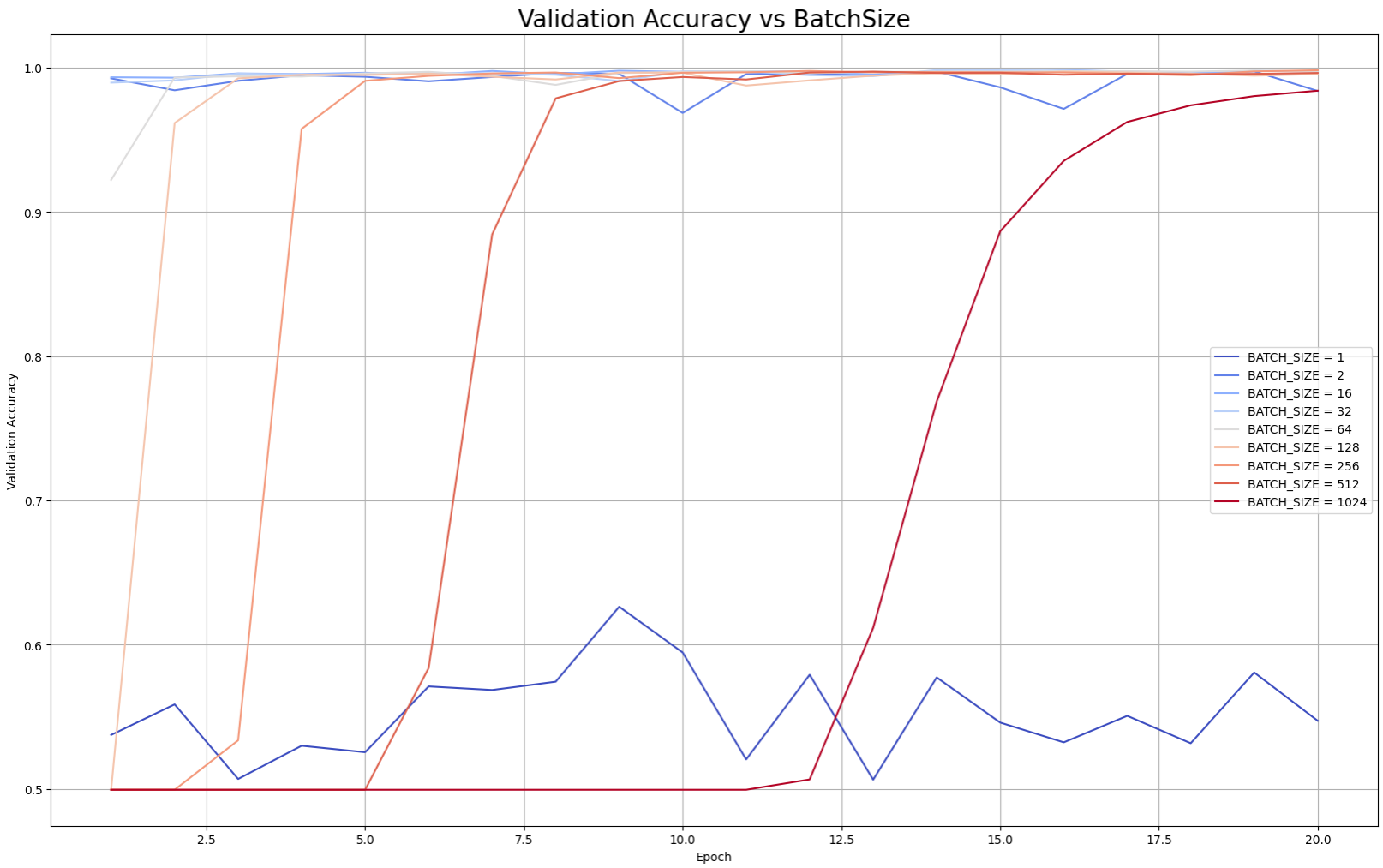




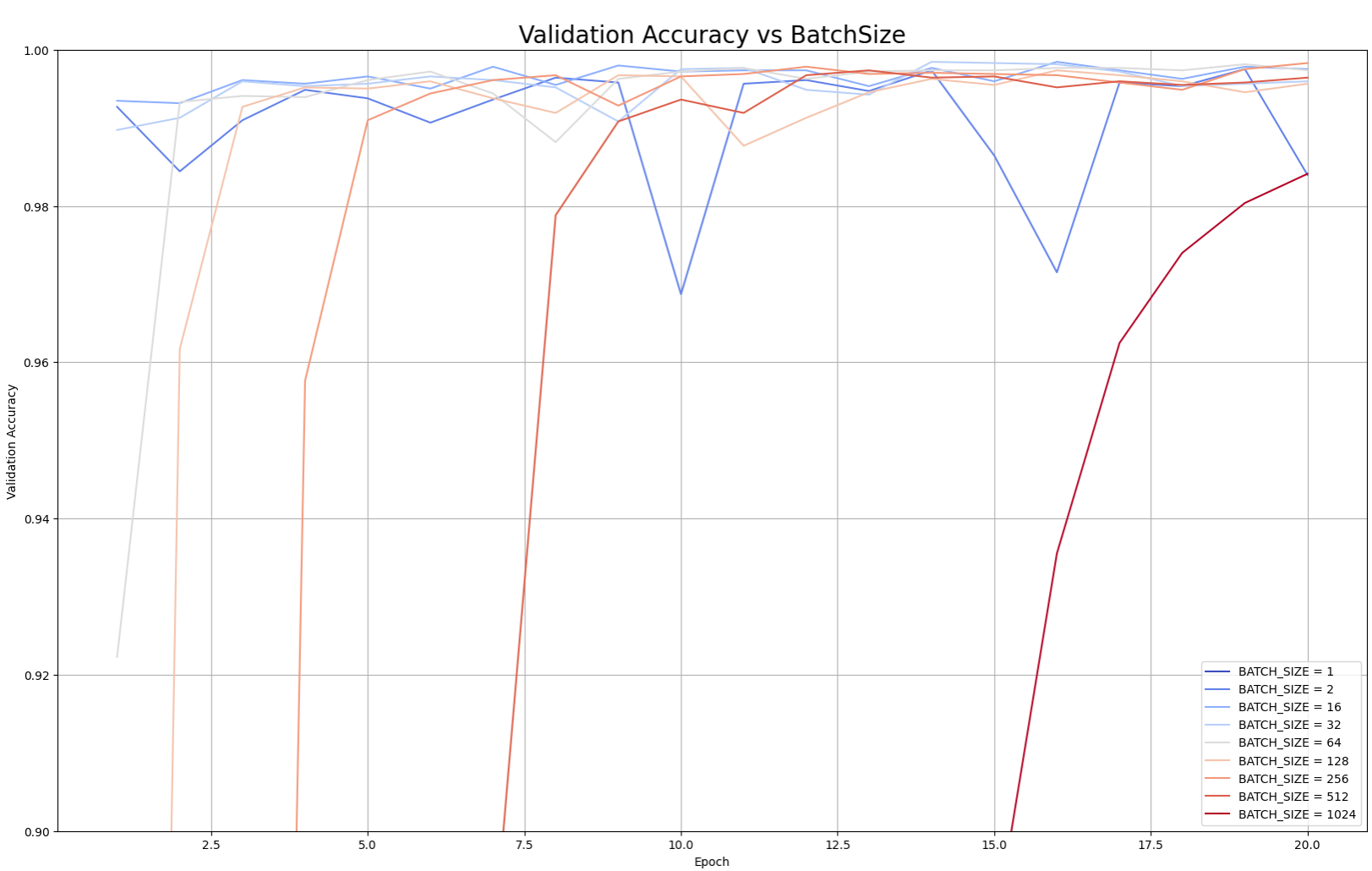




input=128x128
lr=1e-4
epoc=20

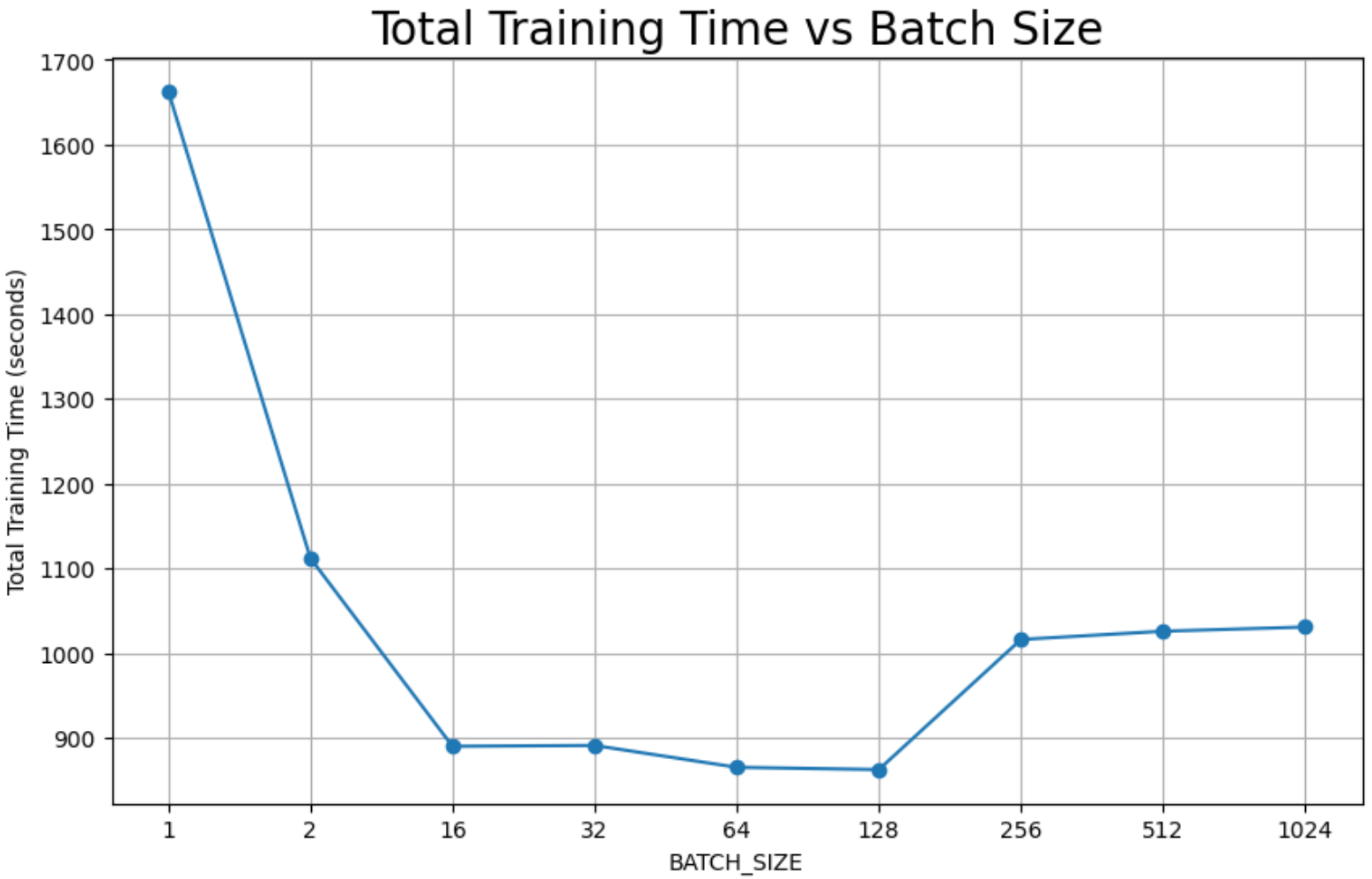


LVX VERITAS VIRTVS



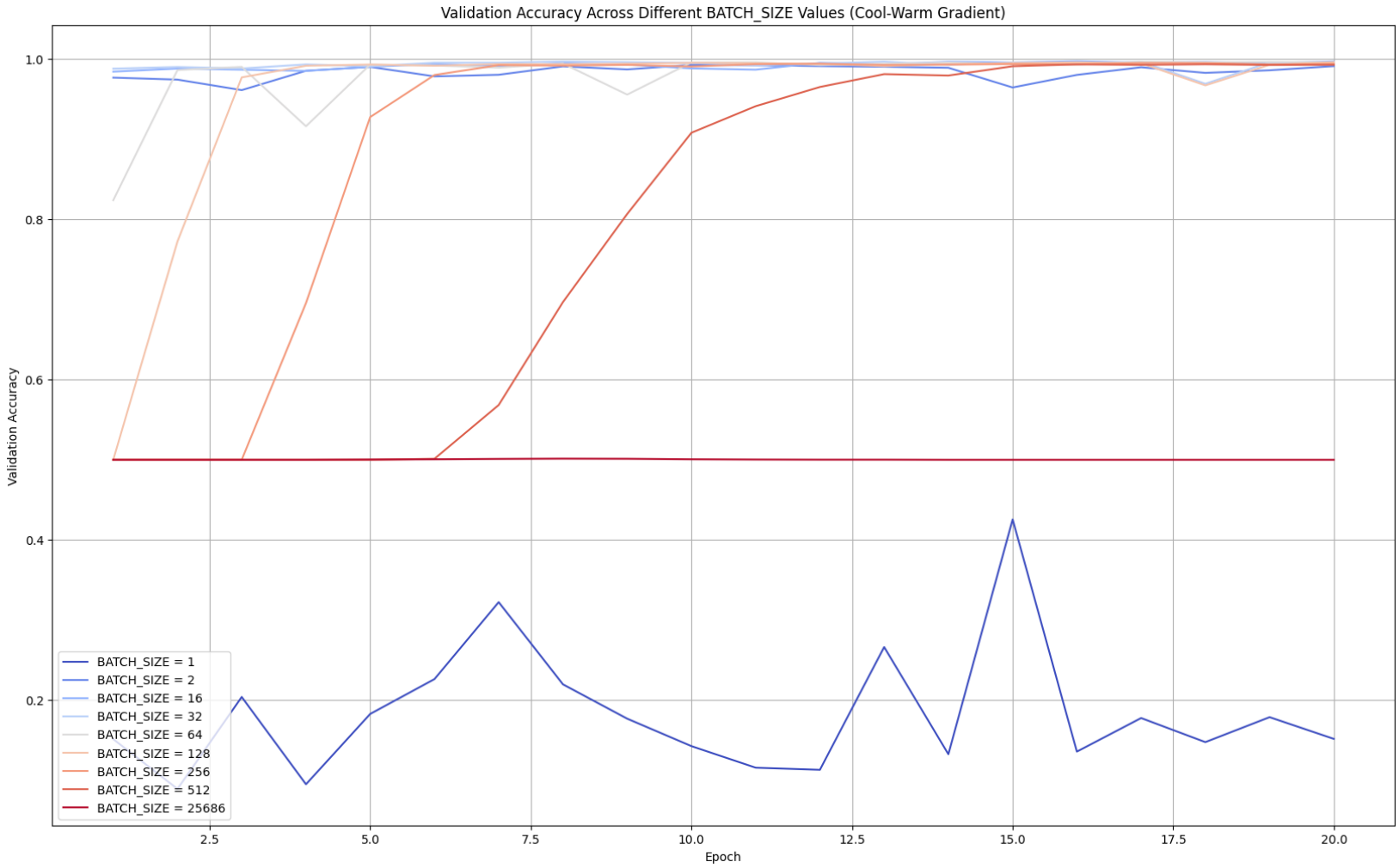
LVX
VERITAS
VIRTVS

input=128x128
lr=1e-4
epoc=20



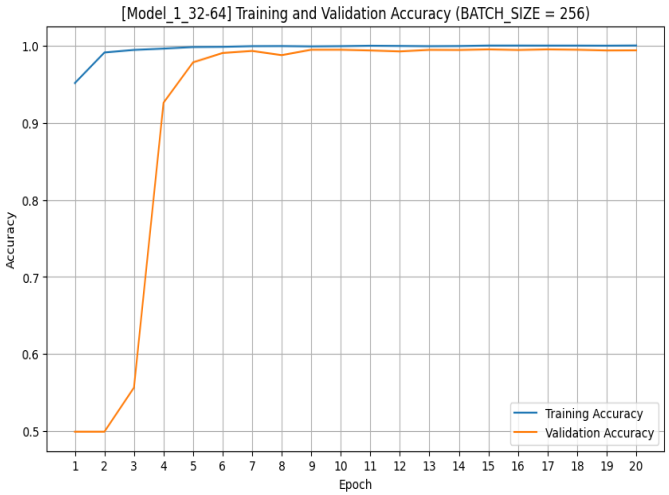
LVX
VERITAS
VIRTVS

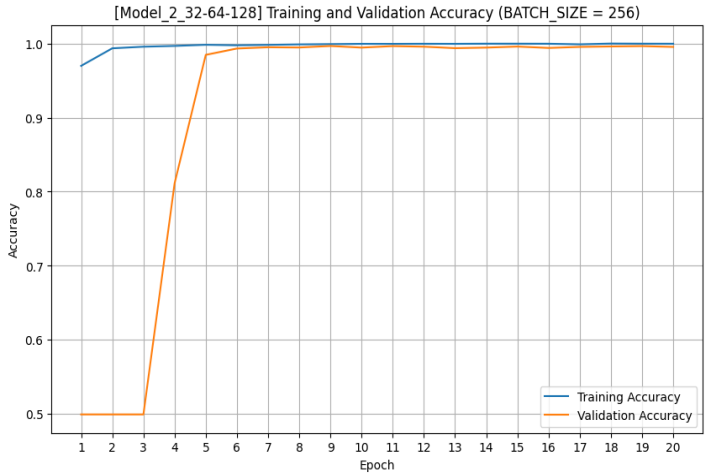
input=32x32

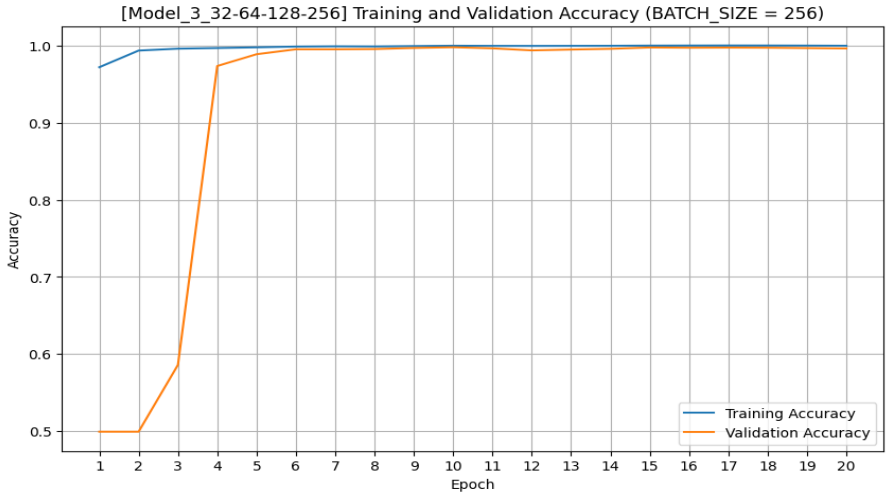

Validation Accuracy Across Different BATCH_SIZE Values (Cool-Warm Gradient)
Validation Accuracy
Epoch
1.0
0.8
0.6
0.4
0.2
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
BATCH_SIZE = 1
BATCH_SIZE = 2
BATCH_SIZE = 16
BATCH_SIZE = 32
BATCH_SIZE = 64
BATCH_SIZE = 128
BATCH_SIZE = 256
BATCH_SIZE = 512
BATCH_SIZE = 25686

4.3 Layers numbers(2,3,4)

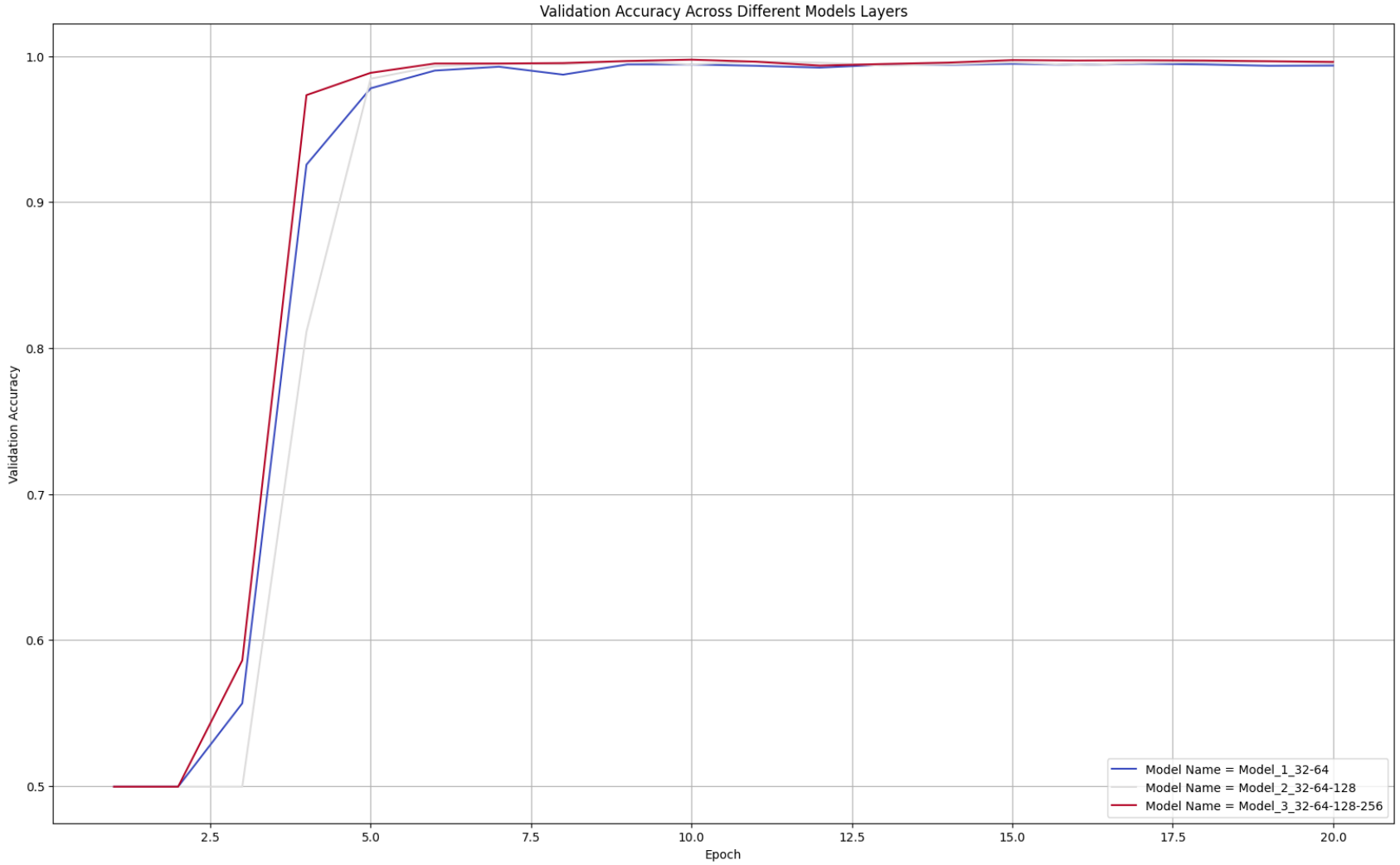
input=128x128
batch_size = 256
lr=1e-4
epoc=20





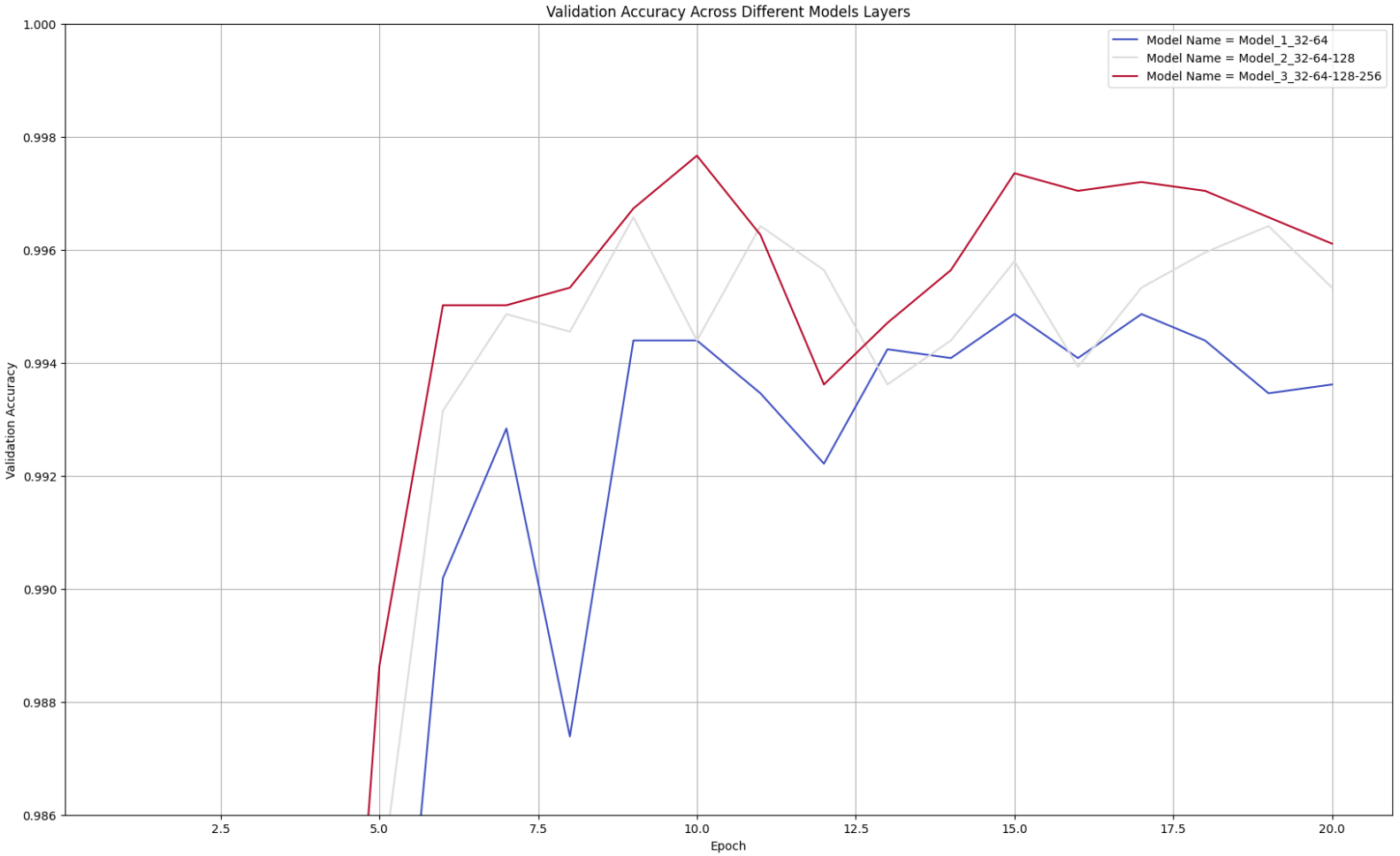


input=128x128
batch_size = 256
lr=1e-4
epoc=20



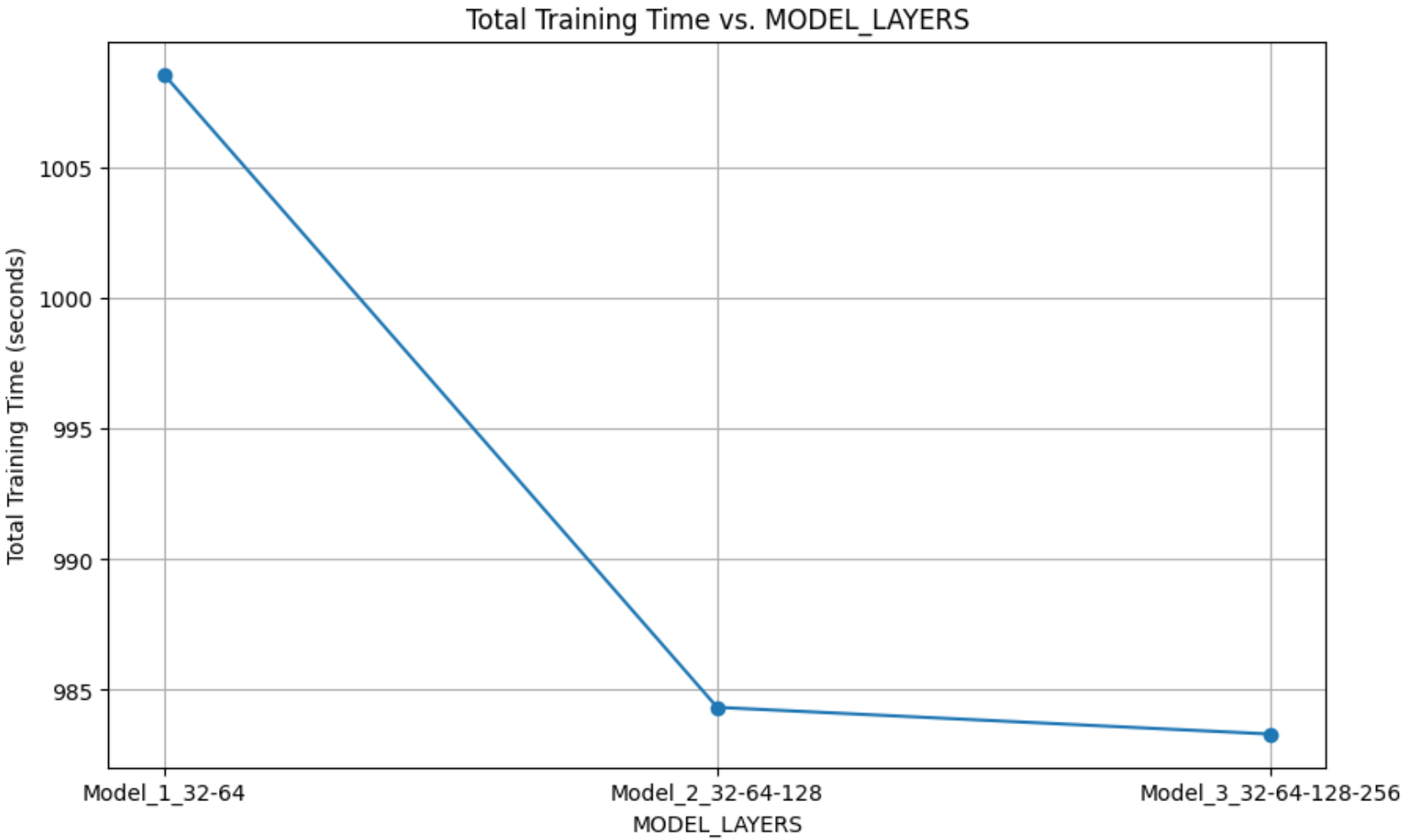
LVX
VERITAS
VIRTVS

input=128x128
batch_size = 256
lr=1e-4
epoc=20



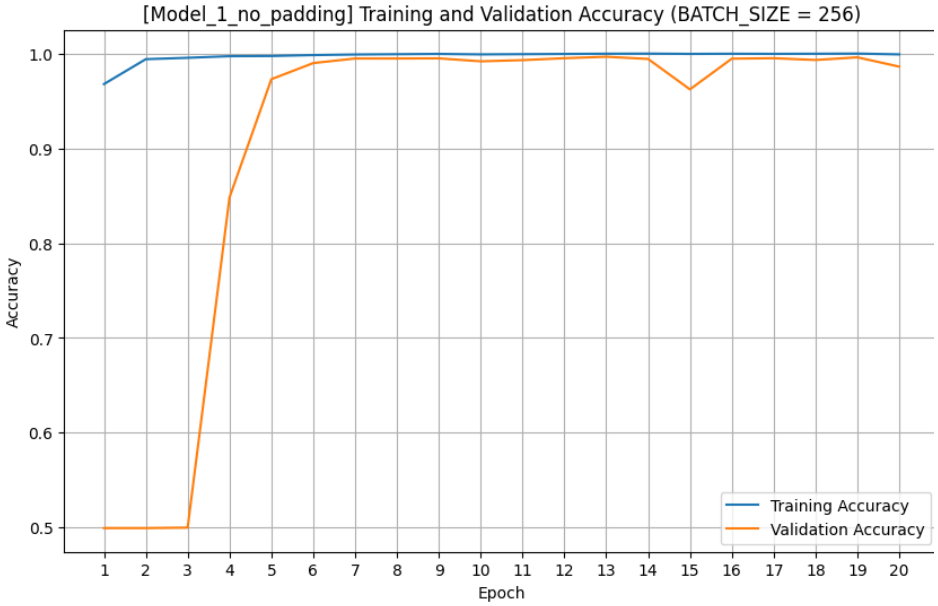
LVX
VERITAS
VIRTVS

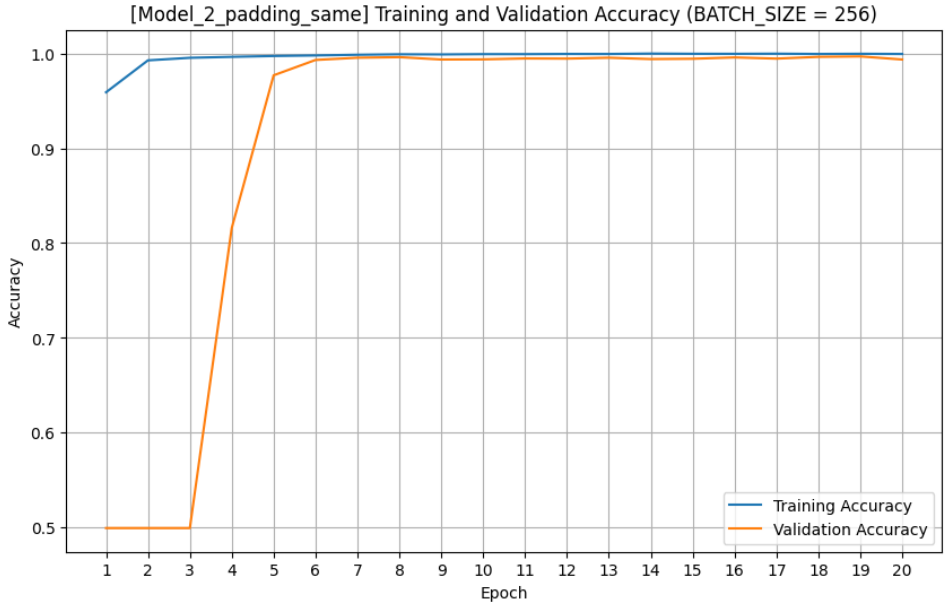
input=128x128
batch_size = 256
lr=1e-4
epoc=20

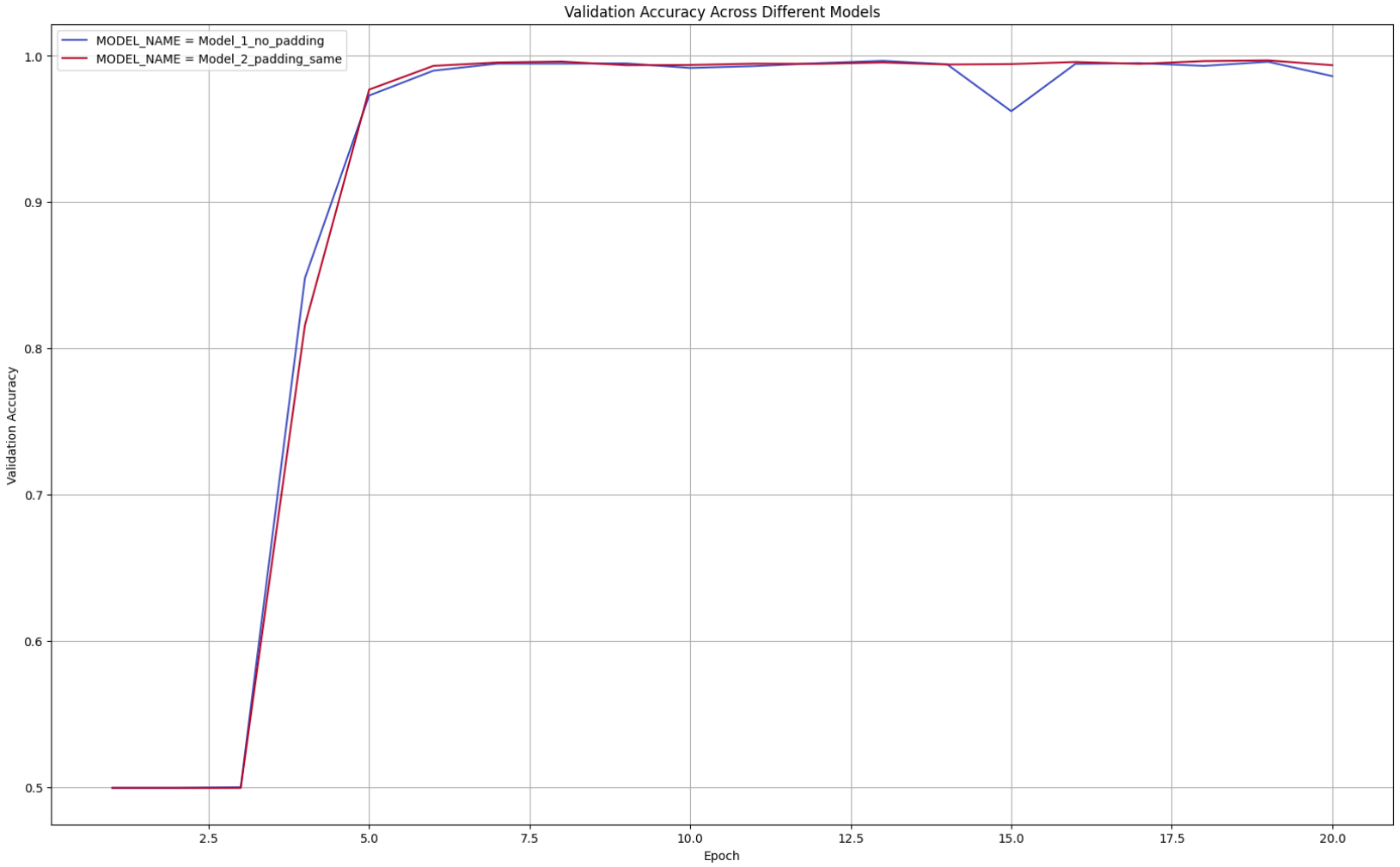


4.4 Model Padding

input=128x128
batch_size = 256
lr=1e-4
epoc=20



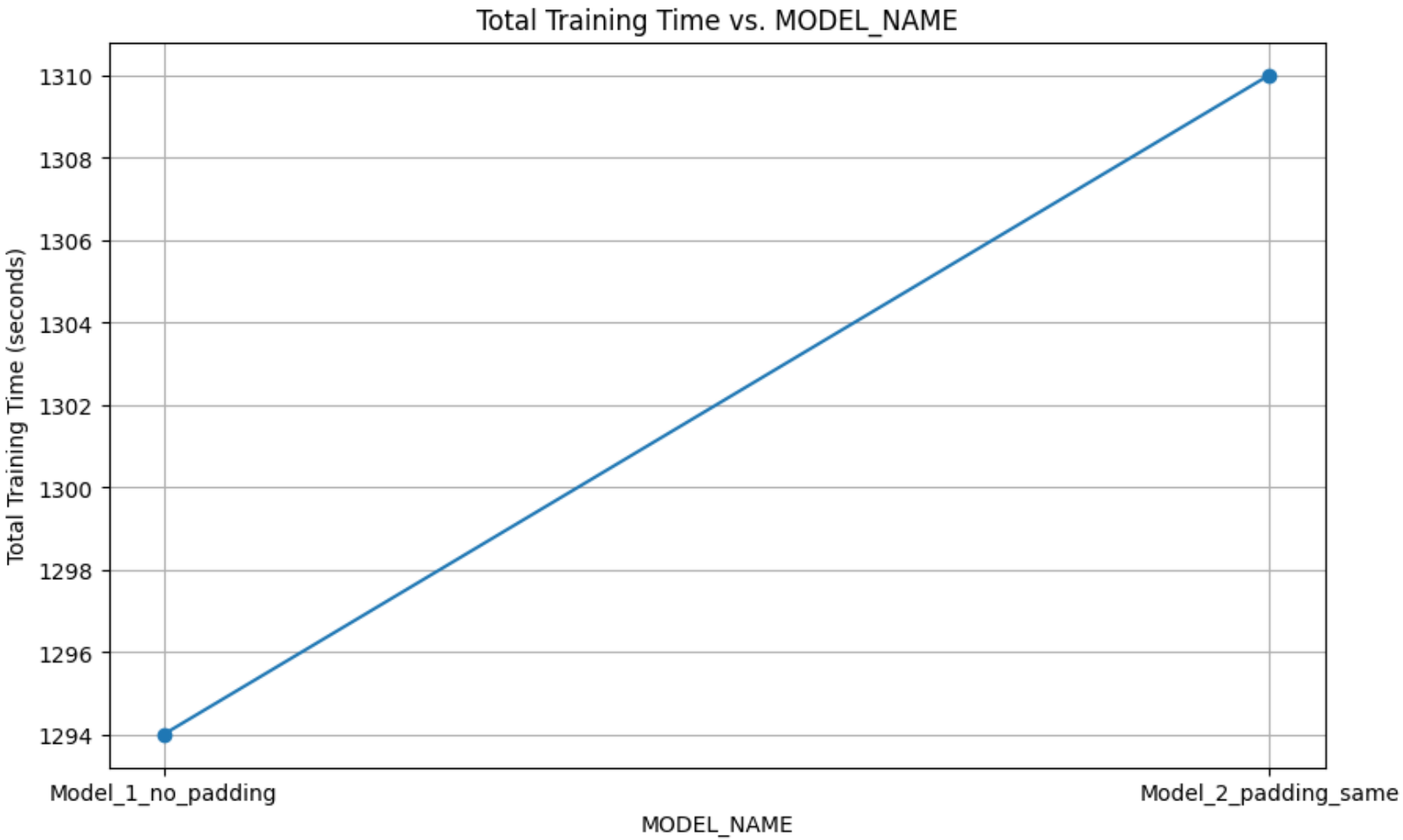




input=128x128
batch_size = 256
lr=1e-4
epoc=20

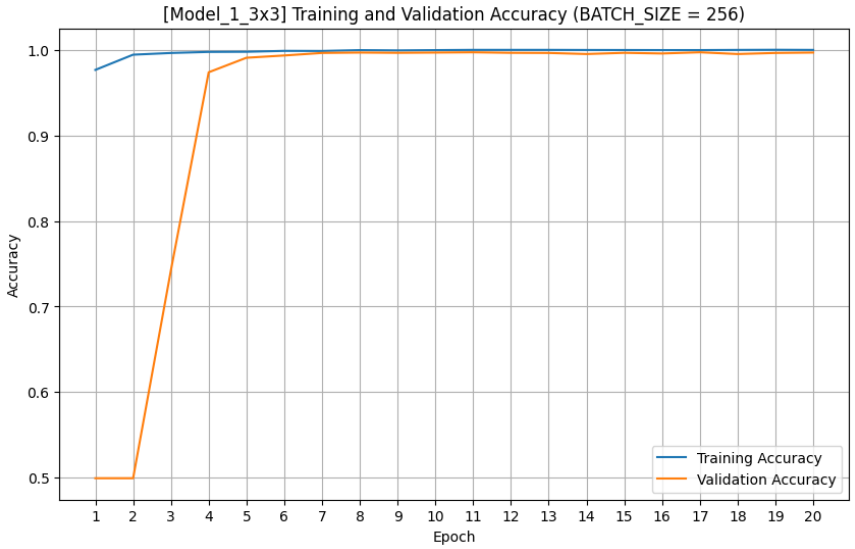
LVX
VERITAS
VIRTVS

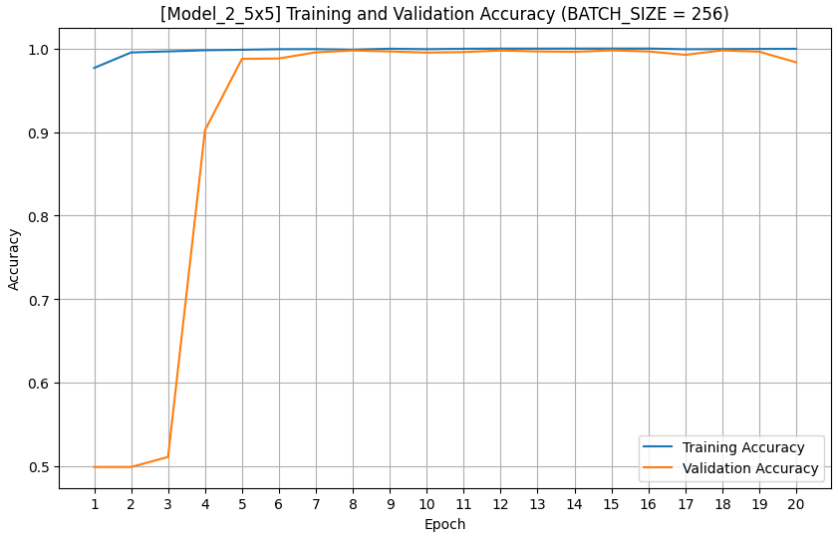
input=128x128
batch_size = 256
lr=1e-4
epoc=20

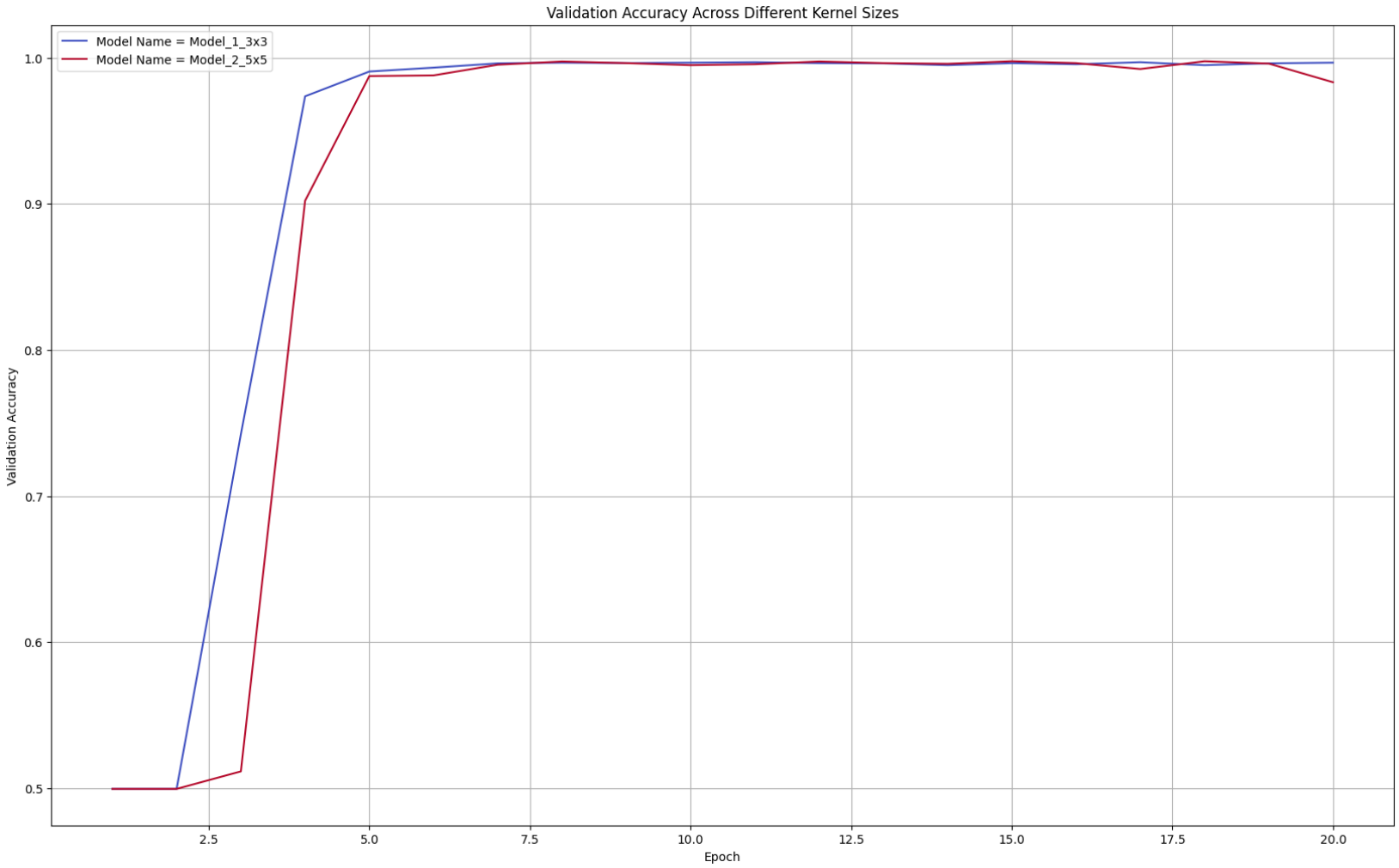


4.5 Kernel Size

input=128x128
batch_size = 256
lr=1e-4
epoc=20

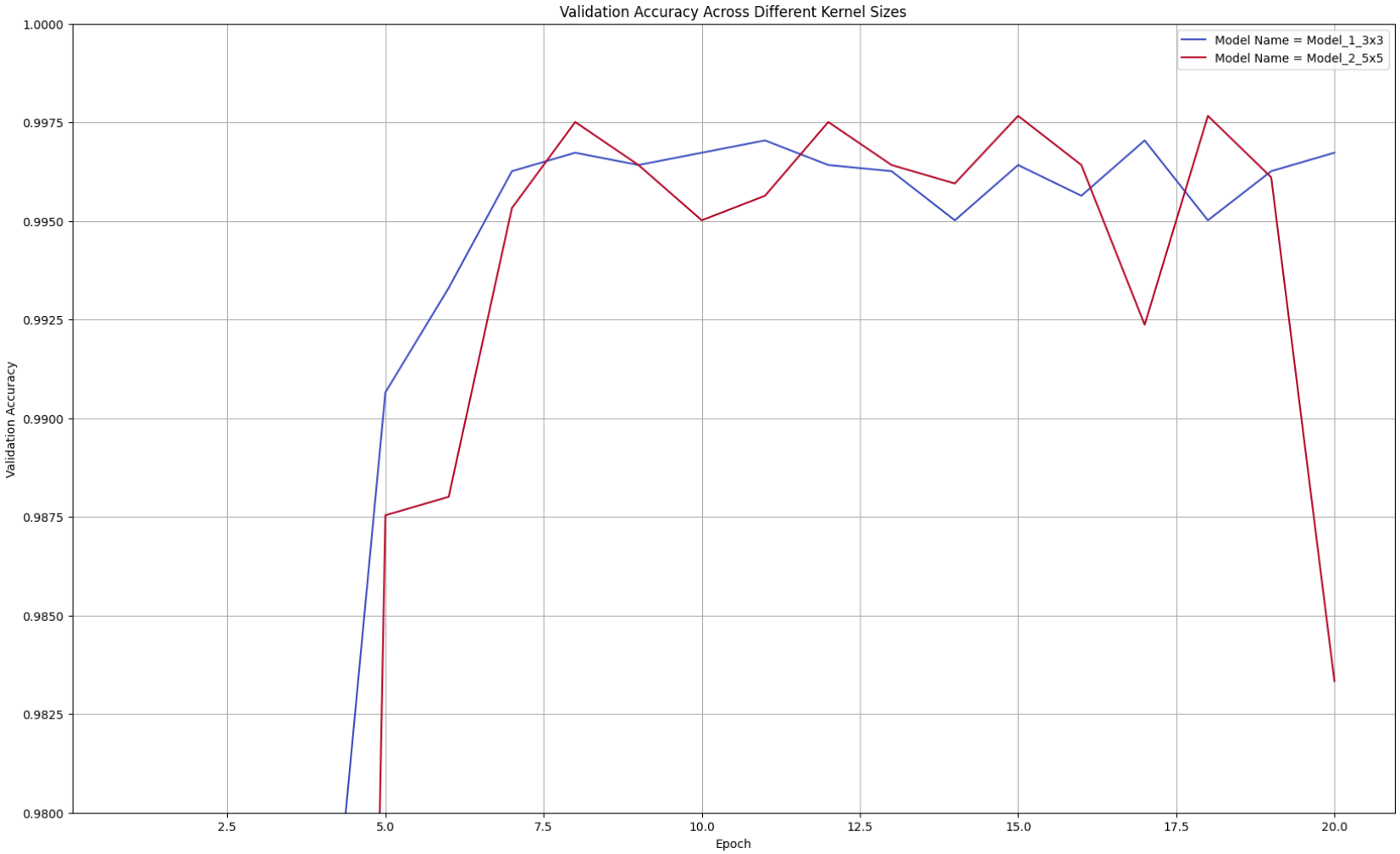




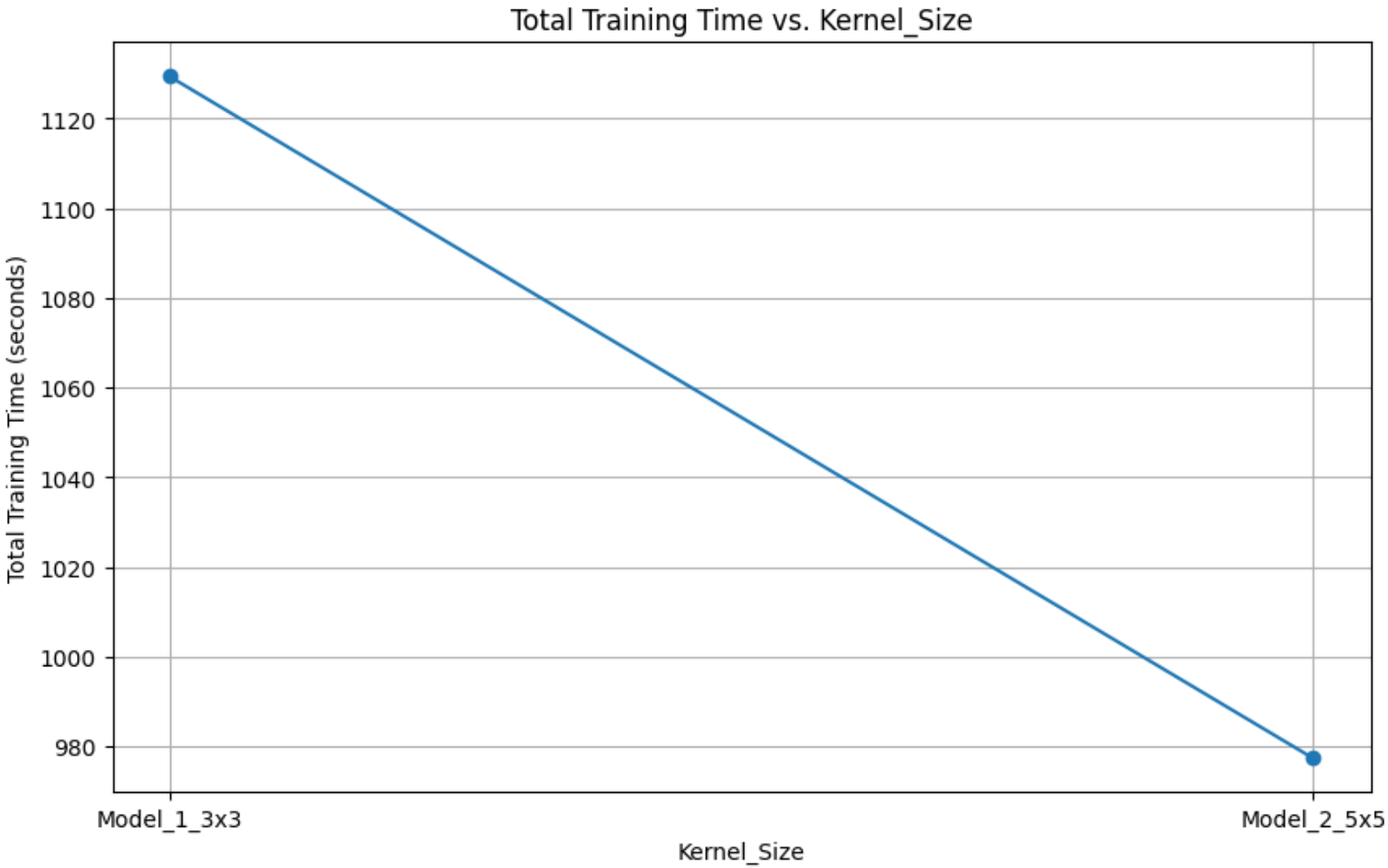


input=128x128
batch_size = 256
lr=1e-4
epoc=20

input=128x128
batch_size = 256
lr=1e-4
epoc=20

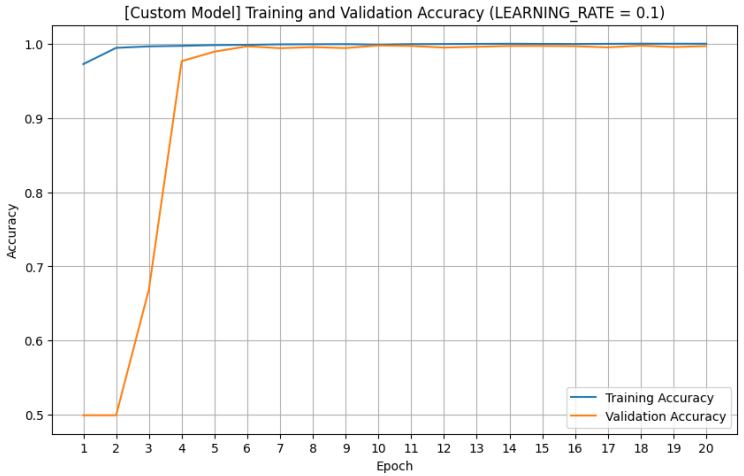


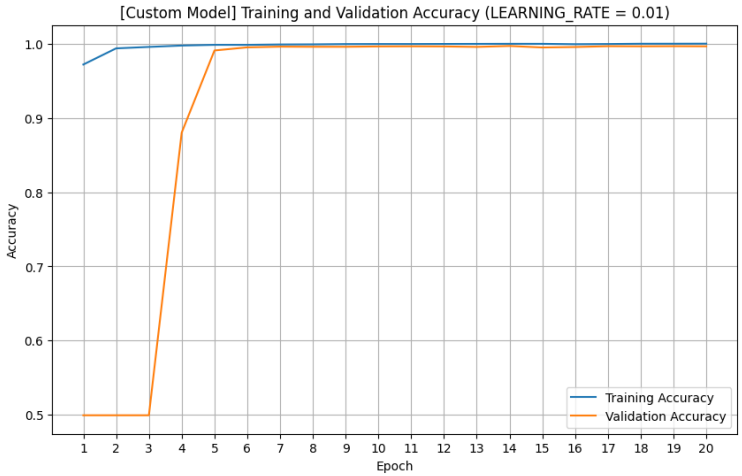
input=128x128
batch_size = 256
lr=1e-4
epoc=20

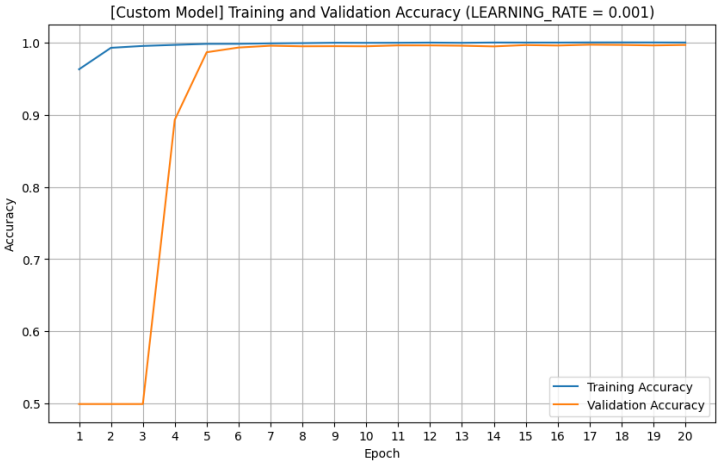


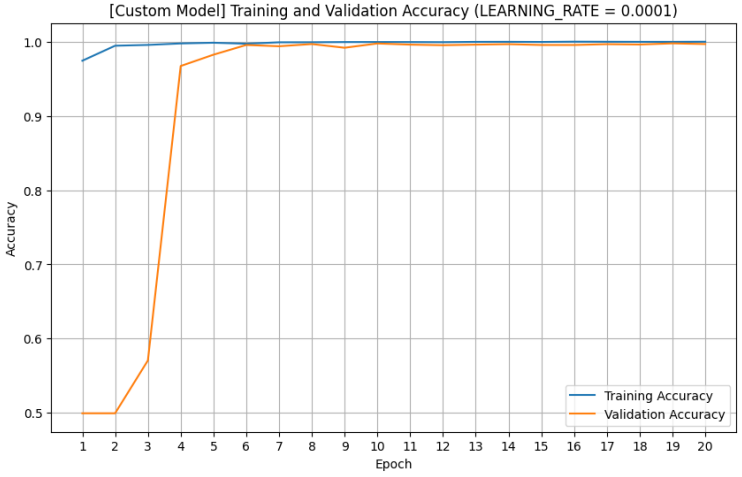
input=128x128
batch_size = 256
epoc=20

4.6 Learning rate

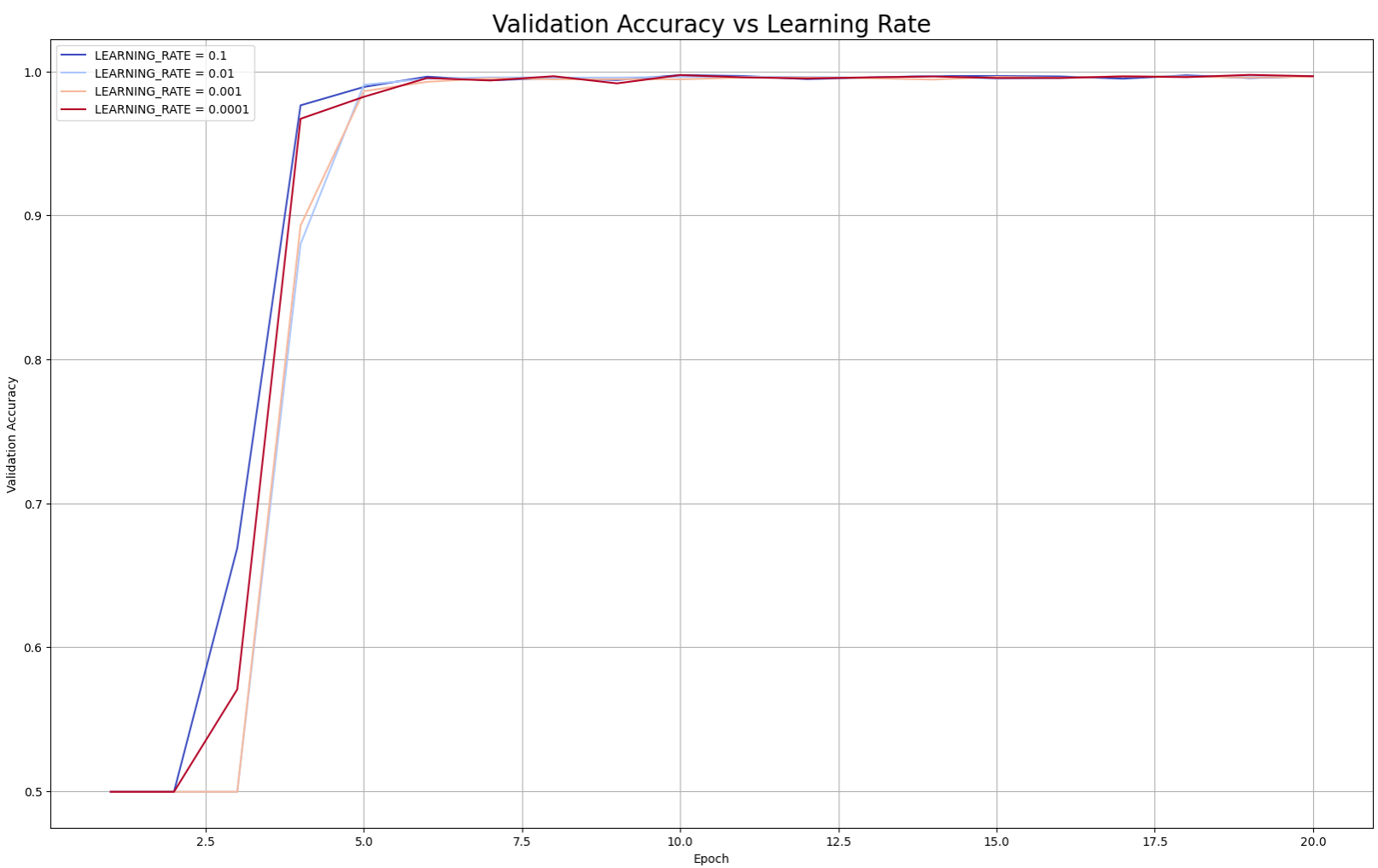








LVX VERITAS VIRTVS



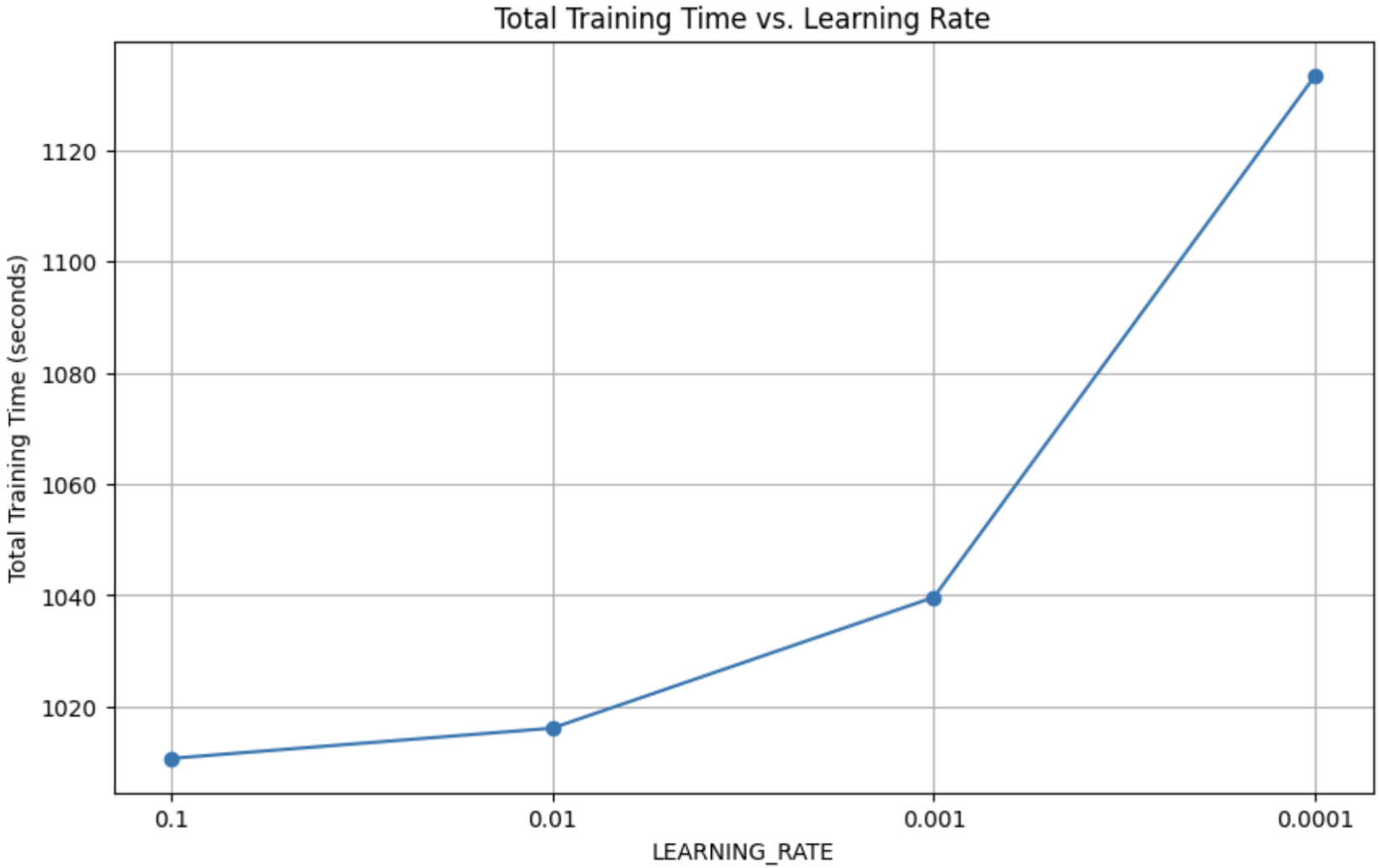
input=128x128
batch_size = 256
epoc=20

LVX
VERITAS
VIRTVS

input=128x128
batch_size = 256
epoc=20

LVX
VERITAS
VIRTVS

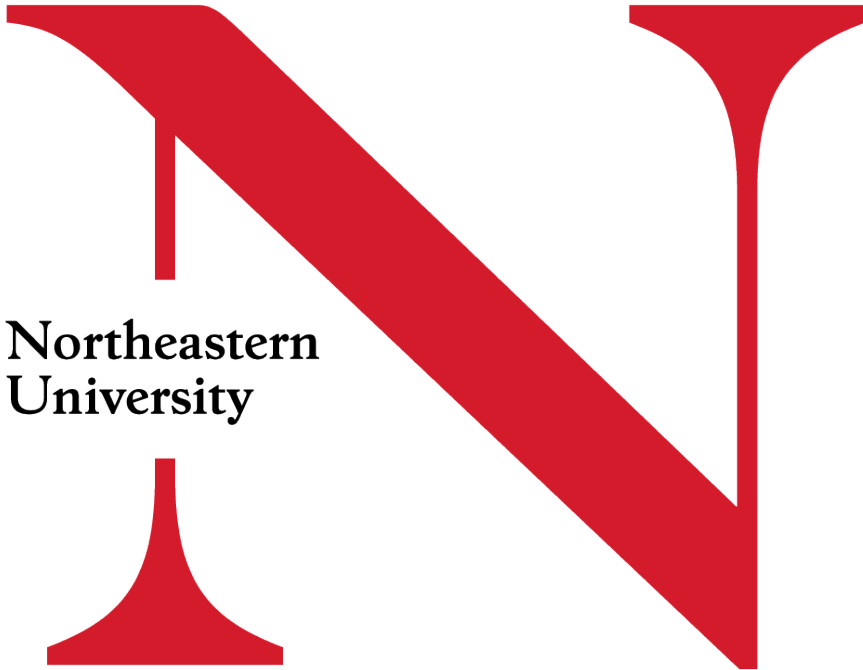
input=128x128
batch_size = 256
epoc=20



Parameter	Larger Impact	Smaller Impact
Input Size	- Increased computation, larger feature maps - Retains more details	- Reduced computation, smaller feature maps - Potential loss of information
Batch Size	- Stable gradients, faster training - Higher memory usage	- Noisier gradients, slower training - Escapes local minima easier
Layer Numbers	- Enhanced feature extraction - Risk of overfitting or vanishing gradients	- Simplified model, easier to train - Limited expressive power
Padding	- Larger feature map size - Retains edge details	- Smaller feature map size - Possible edge information loss
Kernel Size	- Larger receptive field, captures global features - Blurred details	- Smaller receptive field, captures local features - Misses global patterns
Learning Rate	- Faster convergence but unstable - Potential divergence	- Slower convergence but stable - Risk of local minima

5.Transfer Learning

- Resnet 50
- EfficientNetB0
- DenseNet121
- MobileNetV2



5.1 Workflow

Data Preprocessing

- Normalize pixel values to [0, 1].
- Resize images to 224x224.
- Split dataset into train & validation.

Model Fine-tuning

- Load pre-trained model
- Freeze lower layers
- Add top layers
  i. GlobalAveragePooling2D
  ii. Dense(512), ReLU, Batch Normalization
  iii. Dropout(50%)
  iv. Dense(1), Sigmoid

Training and Optimization

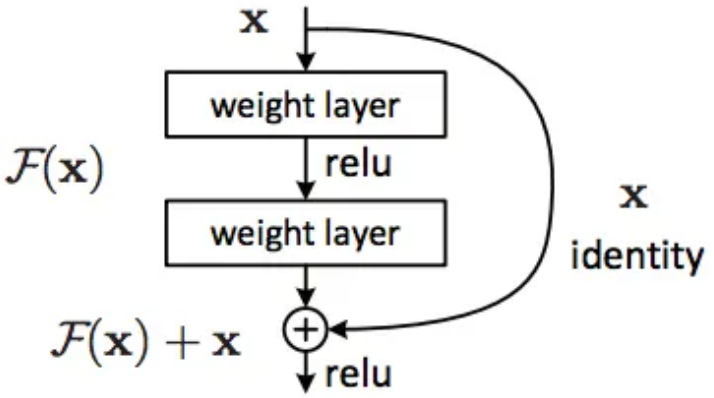
- Adam optimizer(lr=1e-4).
- Loss: Binary Cross-Entropy.
- Metrics: Accuracy
- Callbacks: EarlyStopping and ReduceLROnPlateau.

Evaluation

- Accuracy.
- ROC AUC.
- Confusion Matrix
- FP and FN results

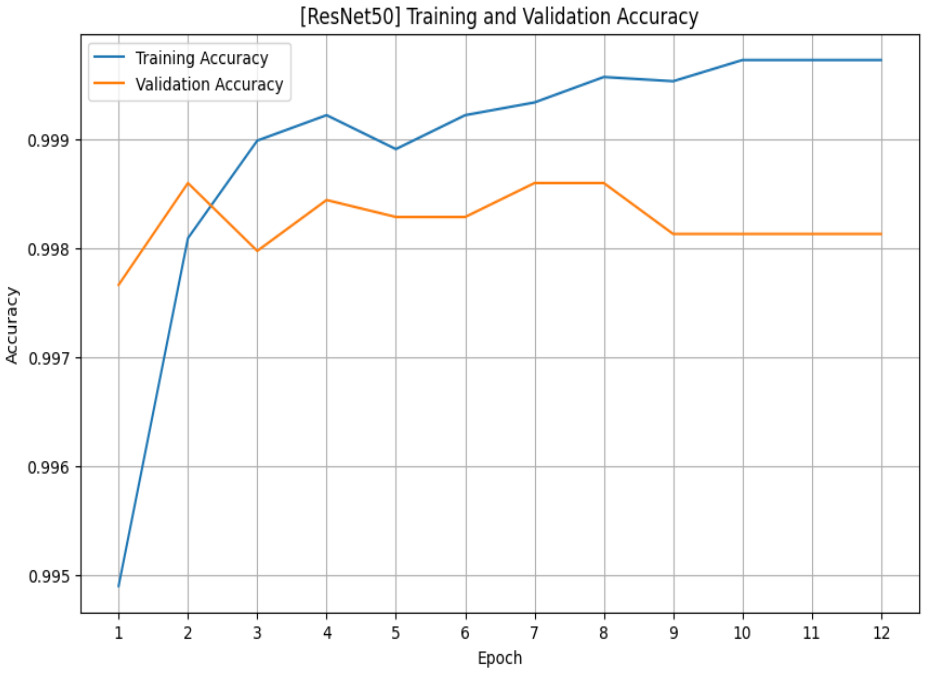
5.2 Resnet50

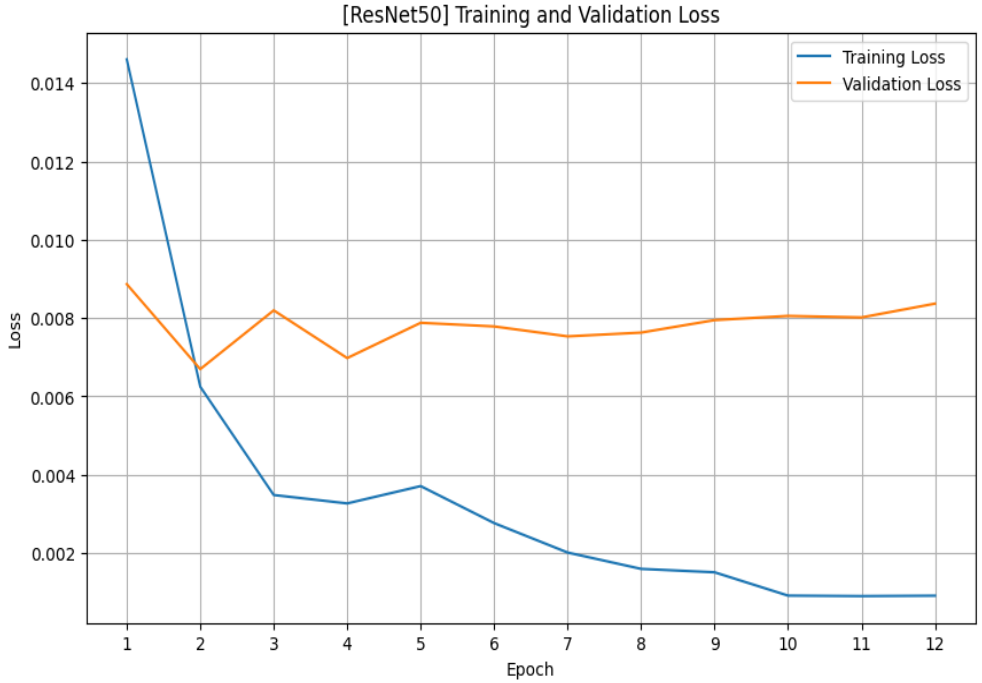
- Microsoft, 2015
- 50-layer architecture solves vanishing gradient issues in deeper networks.
- Skip connection: skips intermediate layers by adding the input directly to the output. It provides a direct path for gradients to flow back to earlier layers.

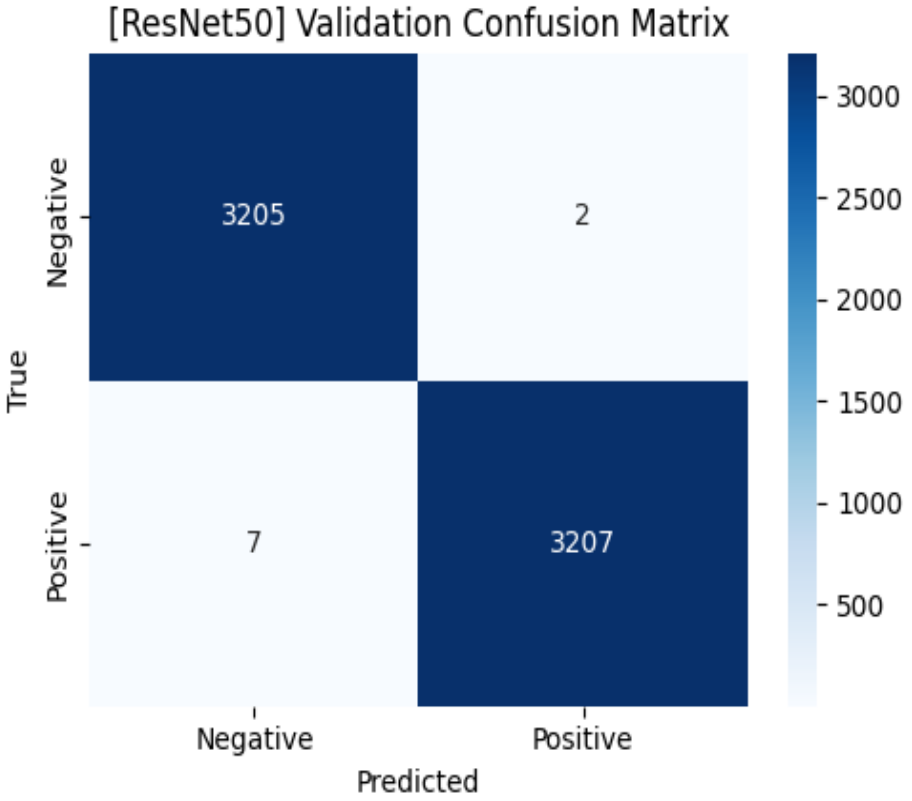


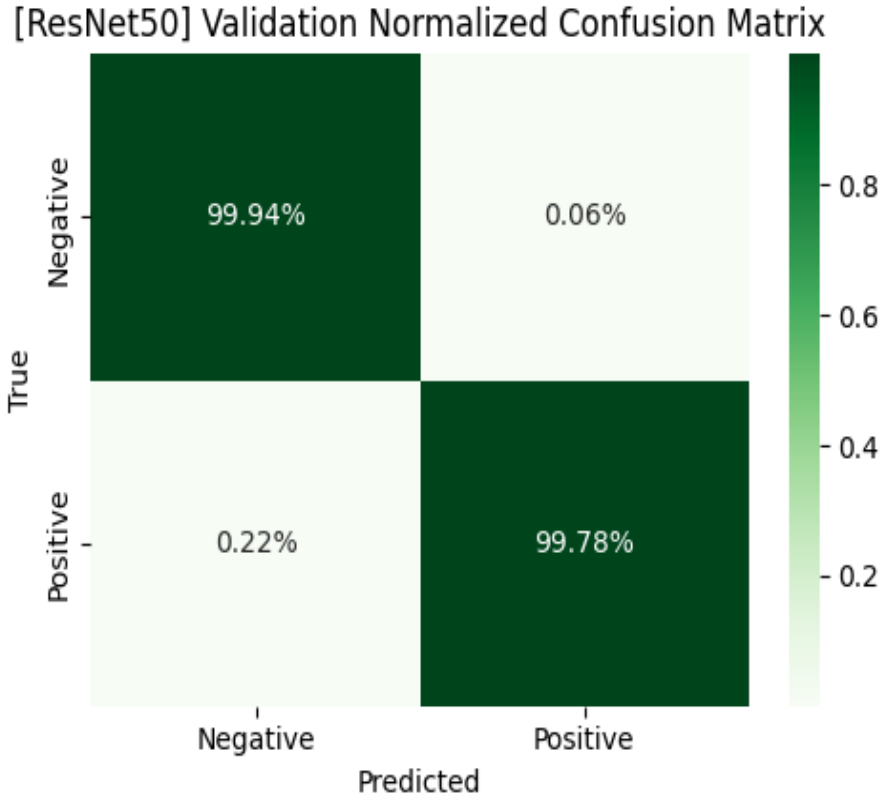
Skip Connection

Layer (type)	Output Shape	Param #
resnet50 (Functional)	(None, 7, 7, 2048)	23,587,712
global_average_pooling2d_6 (GlobalAveragePooling2D)	(None, 2048)	0
dense_16 (Dense)	(None, 512)	1,049,088
batch_normalization_14 (BatchNormalization)	(None, 512)	2,048
dropout_8 (Dropout)	(None, 512)	0
dense_17 (Dense)	(None, 1)	513


[ResNet50] Training and Validation Accuracy
Training Accuracy
Validation Accuracy
Accuracy
Epoch
0.995
0.996
0.997
0.998
0.999
1
2
3
4
5
6
7
8
9
10
11
12

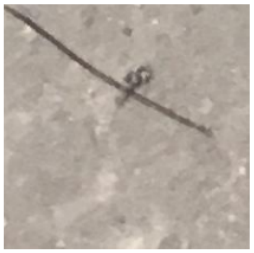

[ResNet50] Training and Validation Loss
Training Loss
Validation Loss
Loss
Epoch
0.002
0.004
0.006
0.008
0.010
0.012
0.014
1
2
3
4
5
6
7
8
9
10
11
12


[ResNet50] Validation Confusion Matrix
Negative
Positive
True
3205
2
7
3207
Negative
Positive
Predicted
3000
2500
2000
1500
1000
500


[ResNet50] Validation Normalized Confusion Matrix
Negative
Positive
True
99.94%
0.06%
0.22%
99.78%
Negative
Positive
Predicted
0.8
0.6
0.4
0.2

False Positives (FP)

True: 0, Pred: 1
Prob: 0.99



True: 0, Pred: 1
Prob: 0.51

False Negatives (FN)

True: 1, Pred: 0
Prob: 0.22

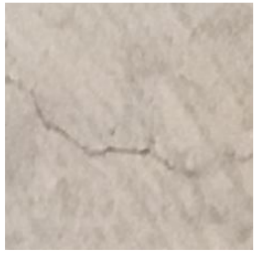
True: 1, Pred: 0
Prob: 0.17

True: 1, Pred: 0
Prob: 0.29

True: 1, Pred: 0
Prob: 0.00

True: 1, Pred: 0
Prob: 0.03

True: 1, Pred: 0
Prob: 0.08



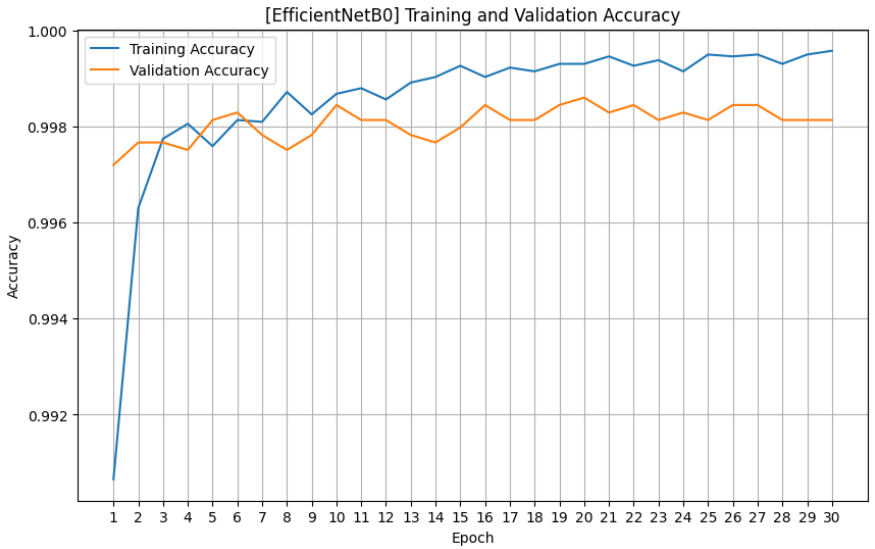
True: 1, Pred: 0
Prob: 0.00

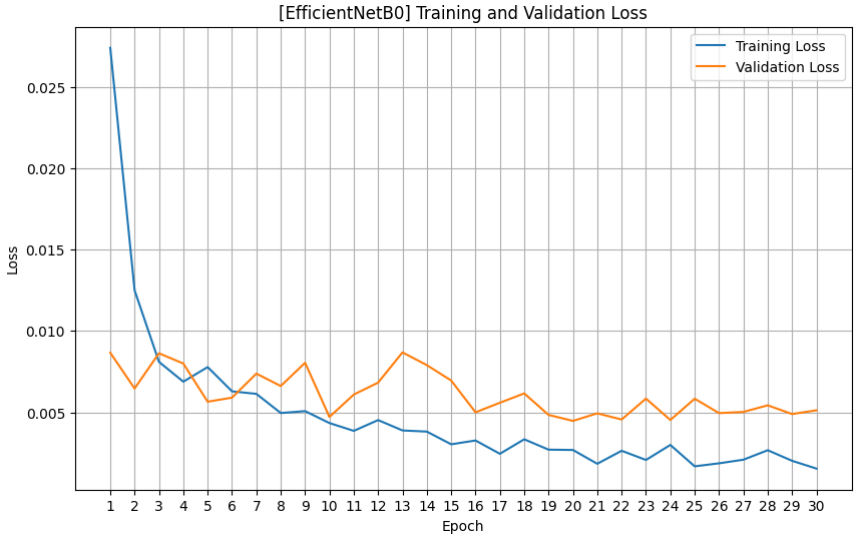
LVX
VERITAS
VIRTVS

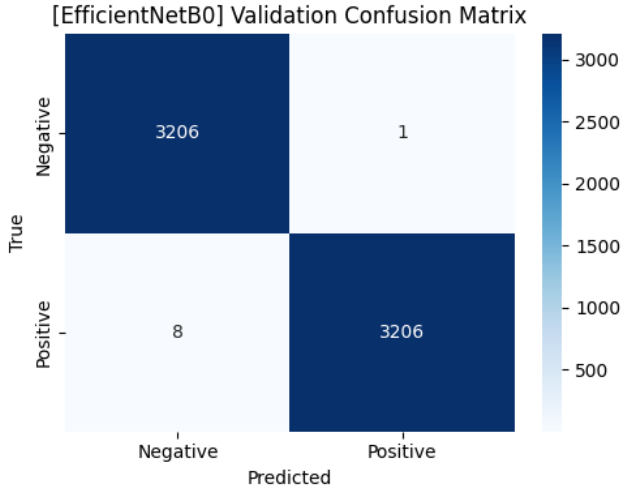
5.3 EfficientNetB0

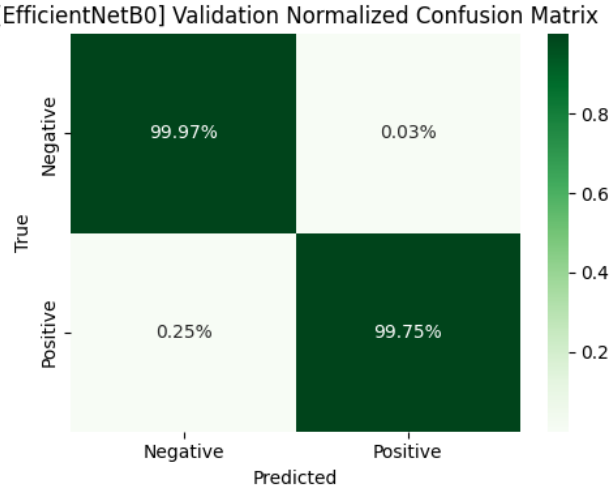
- Google, 2019
- Combines width, depth, and resolution optimally for high accuracy.
- Balances accuracy and efficiency, reduces parameters and computation.

Layer (type)	Output Shape	Param #
efficientnetb0 (Functional)	(None, 7, 7, 1280)	4,049,571
global_average_pooling2d_7 (GlobalAveragePooling2D)	(None, 1280)	0
dense_18 (Dense)	(None, 512)	655,872
batch_normalization_15 (BatchNormalization)	(None, 512)	2,048
dropout_9 (Dropout)	(None, 512)	0
dense_19 (Dense)	(None, 1)	513


[EfficientNetB0] Training and Validation Accuracy
Training Accuracy
Validation Accuracy
Accuracy
Epoch


[EfficientNetB0] Training and Validation Loss
Training Loss
Validation Loss
Loss
Epoch


[EfficientNetB0] Validation Confusion Matrix
3206
1
8
3206
Negative
Positive
True
Predicted


[EfficientNetB0] Validation Normalized Confusion Matrix
99.97%
0.03%
0.25%
99.75%
Negative
Positive
True
Predicted

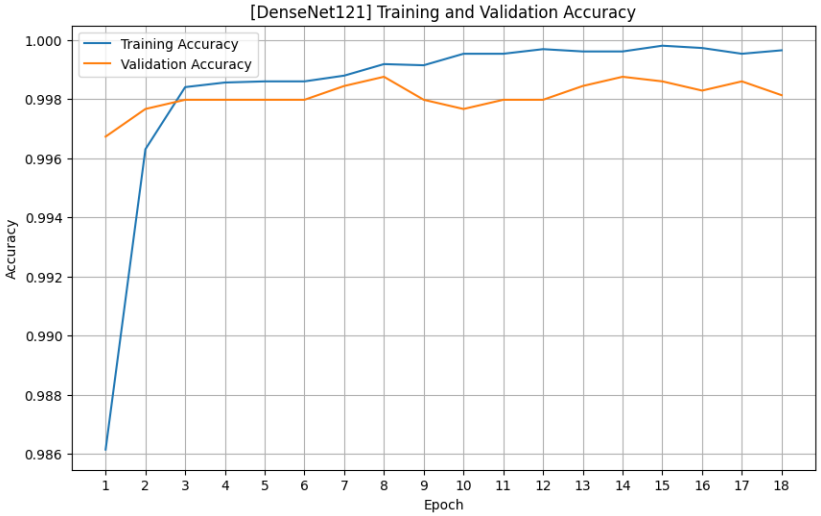
5.4 DenseNet121

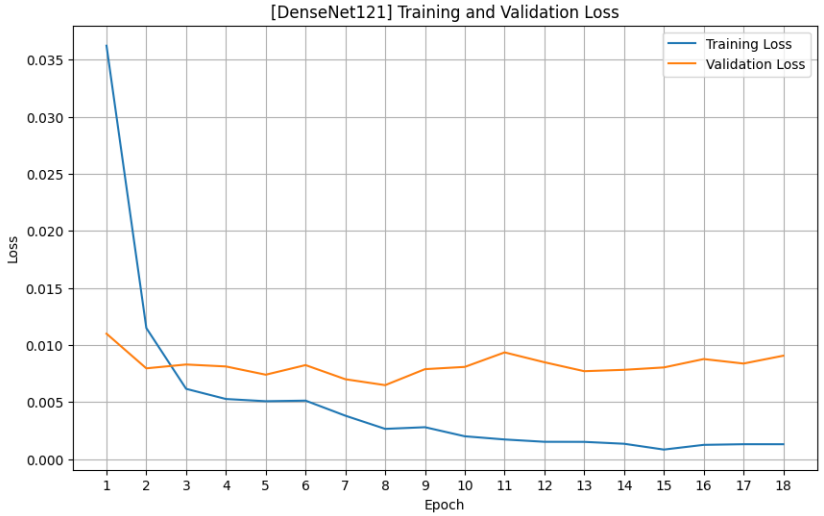
- Cornell University, 2017.

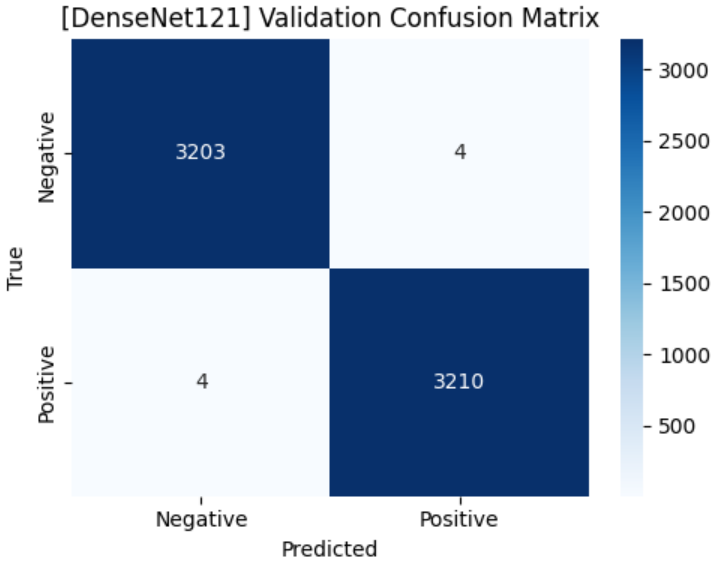
- Each layer connects to all previous layers, enabling feature reuse and efficient gradient flow.

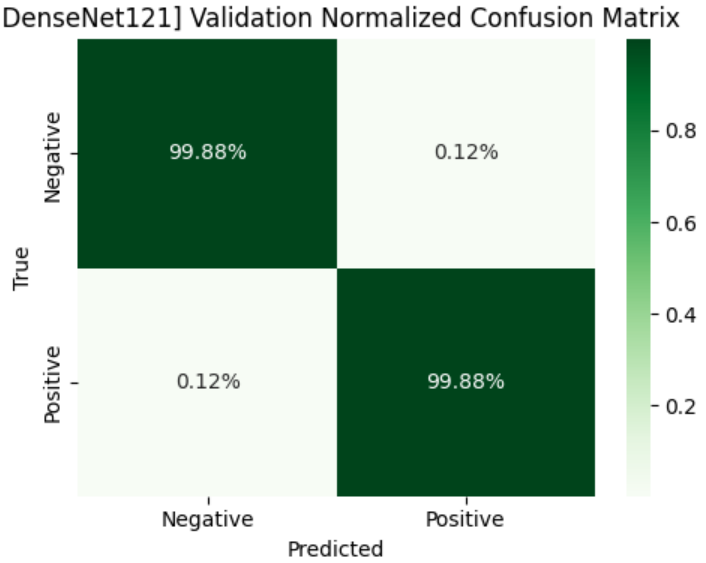
- Reduces parameters through feature reuse, improves gradient propagation.

Layer (type)	Output Shape	Param #
densenet121 (Functional)	(None, 7, 7, 1024)	7,037,504
global_average_pooling2d_8 (GlobalAveragePooling2D)	(None, 1024)	0
dense_20 (Dense)	(None, 512)	524,800
batch_normalization_16 (BatchNormalization)	(None, 512)	2,048
dropout_10 (Dropout)	(None, 512)	0
dense_21 (Dense)	(None, 1)	513


[DenseNet121] Training and Validation Accuracy
Training Accuracy
Validation Accuracy
Accuracy
Epoch


[DenseNet121] Training and Validation Loss
Training Loss
Validation Loss
Loss
Epoch


[DenseNet121] Validation Confusion Matrix
Negative
Positive
True
3203
4
4
3210
Negative
Positive
Predicted


[DenseNet121] Validation Normalized Confusion Matrix
Negative
Positive
True
99.88%
0.12%
0.12%
99.88%
Negative
Positive
Predicted

5.5 MobileNetV2

- Google, 2018, for mobile-first applications.

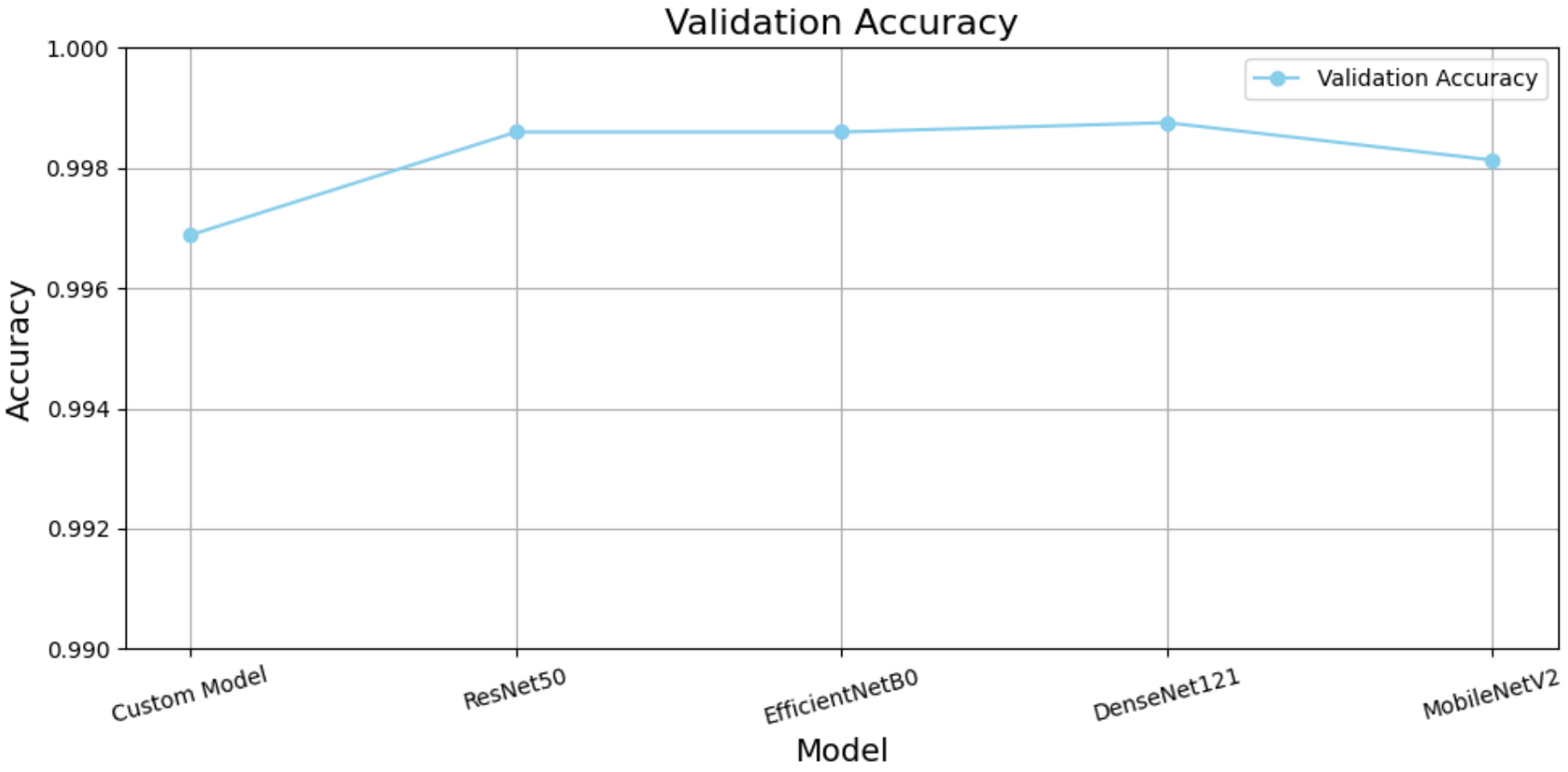
- Uses depthwise separable convolutions and inverted residuals to minimize computation.

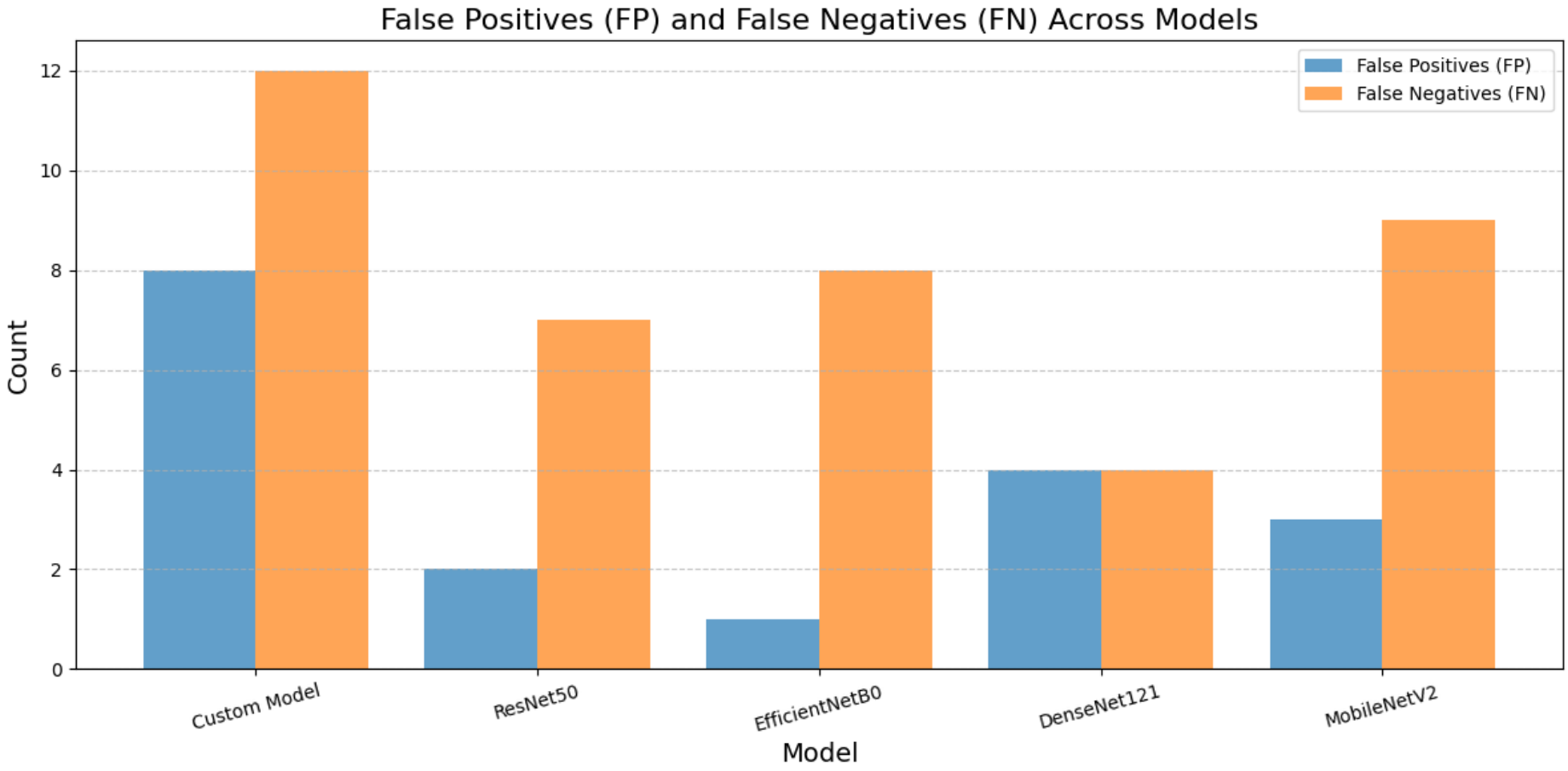
- Highly efficient for mobile devices.

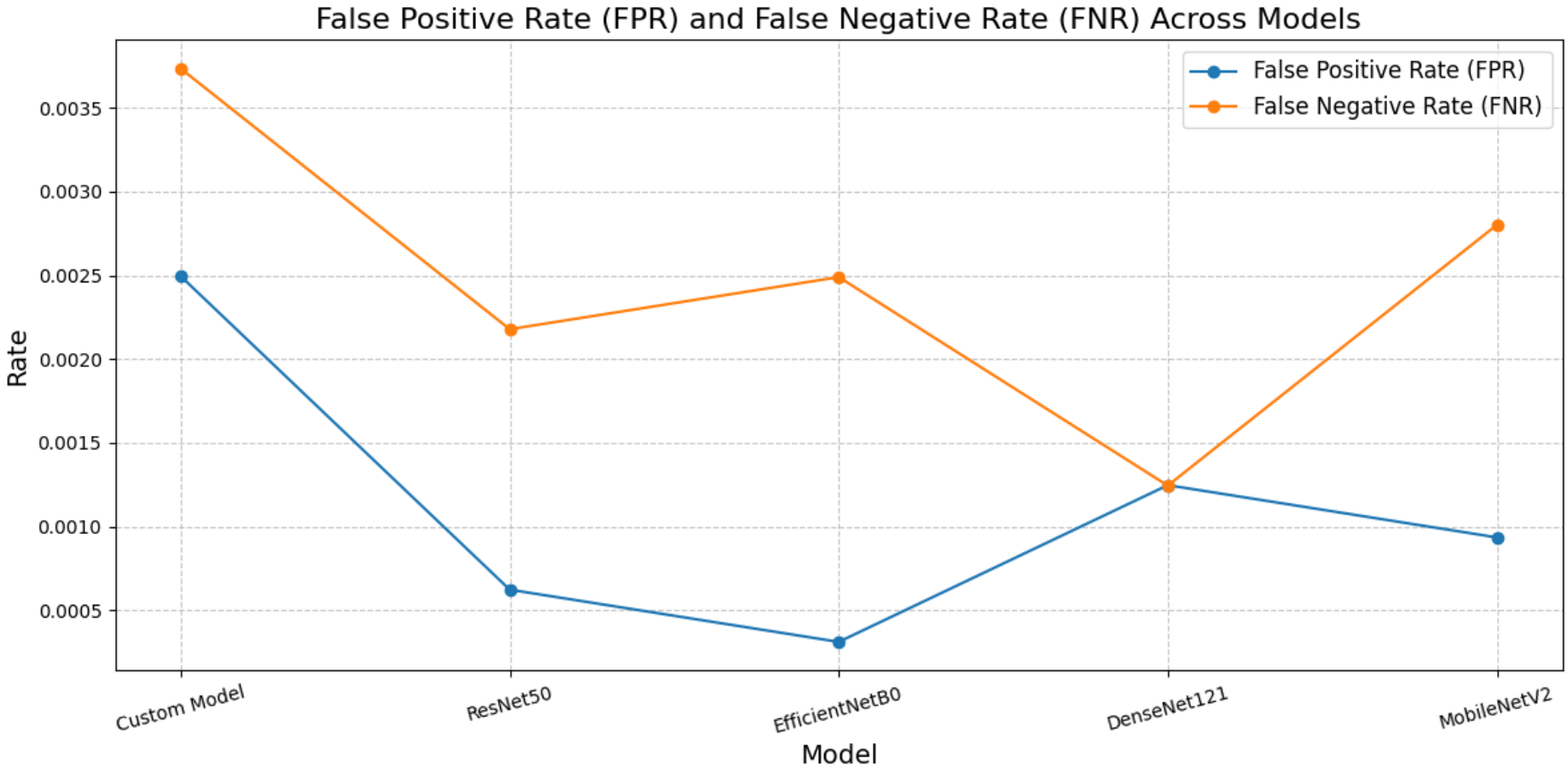
Layer (type)	Output Shape	Param #
mobilenetv2_1.00_224 (Functional)	(None, 7, 7, 1280)	2,257,984
global_average_pooling2d_9 (GlobalAveragePooling2D)	(None, 1280)	0
dense_22 (Dense)	(None, 512)	655,872
batch_normalization_17 (BatchNormalization)	(None, 512)	2,048
dropout_11 (Dropout)	(None, 512)	0
dense_23 (Dense)	(None, 1)	513

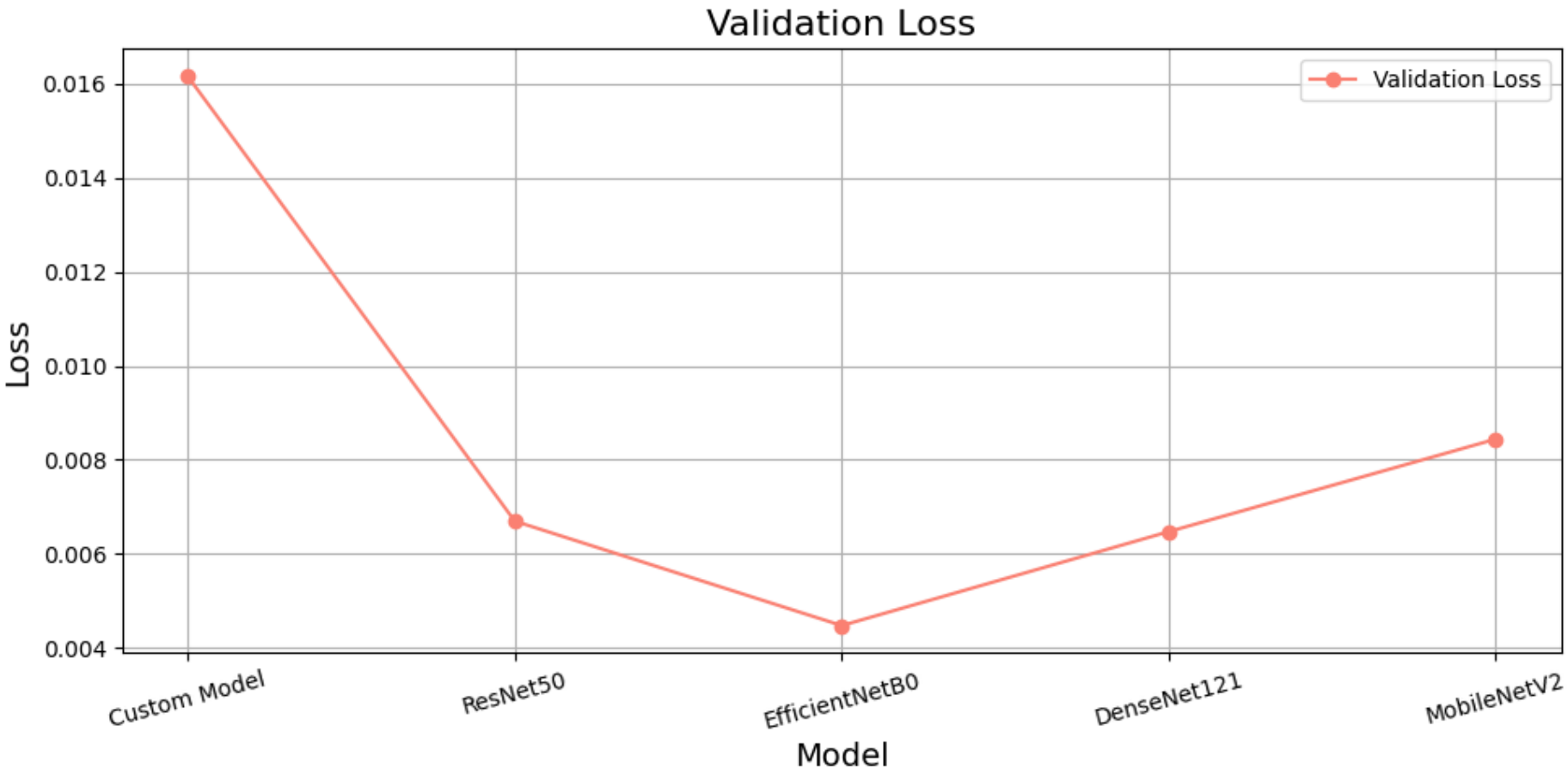
6. Model Comparison

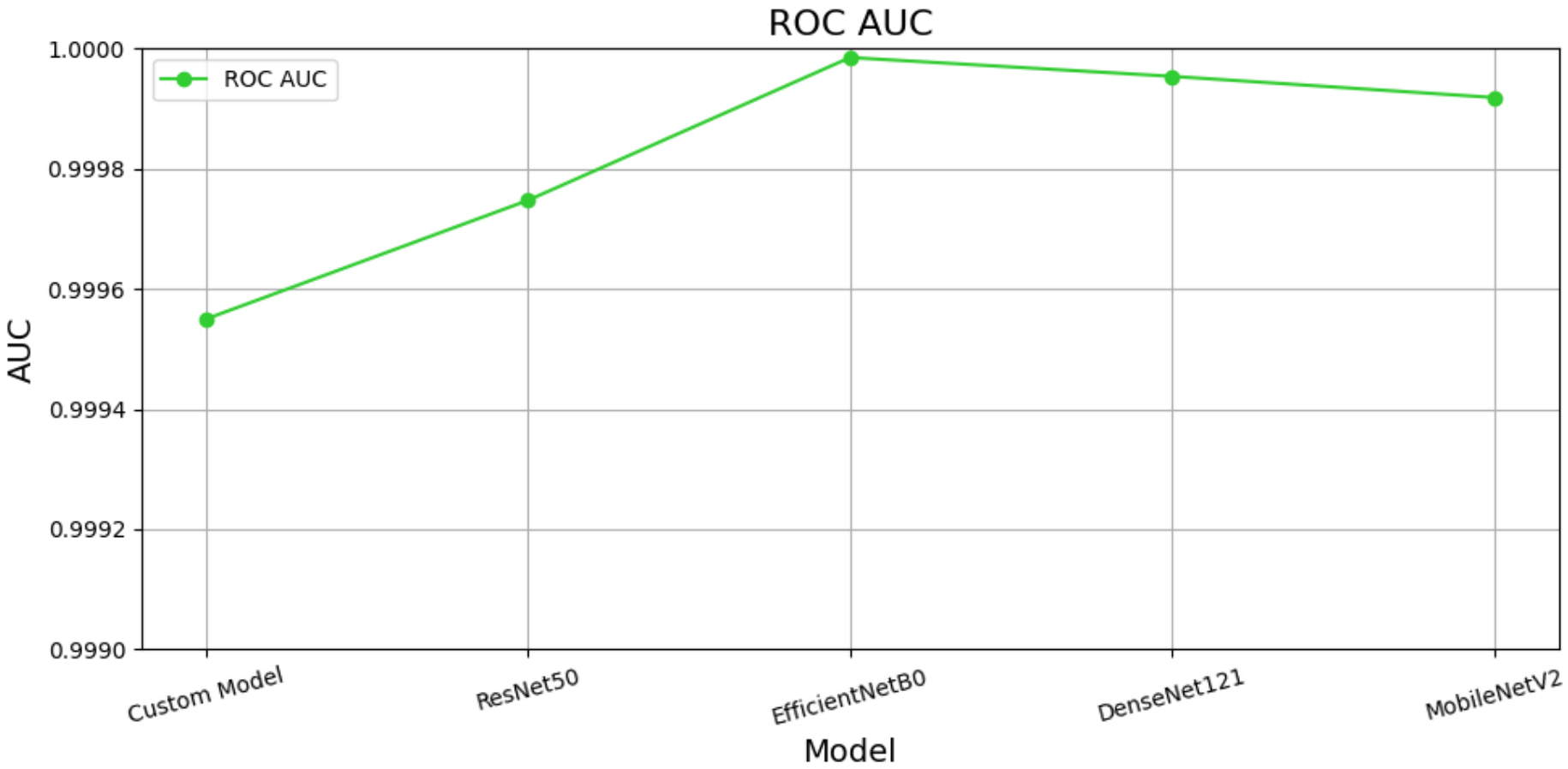
	Model	Validation Accuracy	Validation Loss	ROC AUC	FP	FN	FPR	FNR	Avg Epoch Time (s)	Epochs	Total Time (s)
0	Custom Model	0.996885	0.016159	0.999550	8	12	0.002495	0.003734	50	19	953
1	ResNet50	0.998598	0.006696	0.999747	2	7	0.000624	0.002178	73	12	884
2	EfficientNetB0	0.998598	0.004474	0.999985	1	8	0.000312	0.002489	63	30	1891
3	DenseNet121	0.998754	0.006473	0.999954	4	4	0.001247	0.001245	71	18	1278
4	MobileNetV2	0.998131	0.008441	0.999919	3	9	0.000935	0.002800	58	12	700

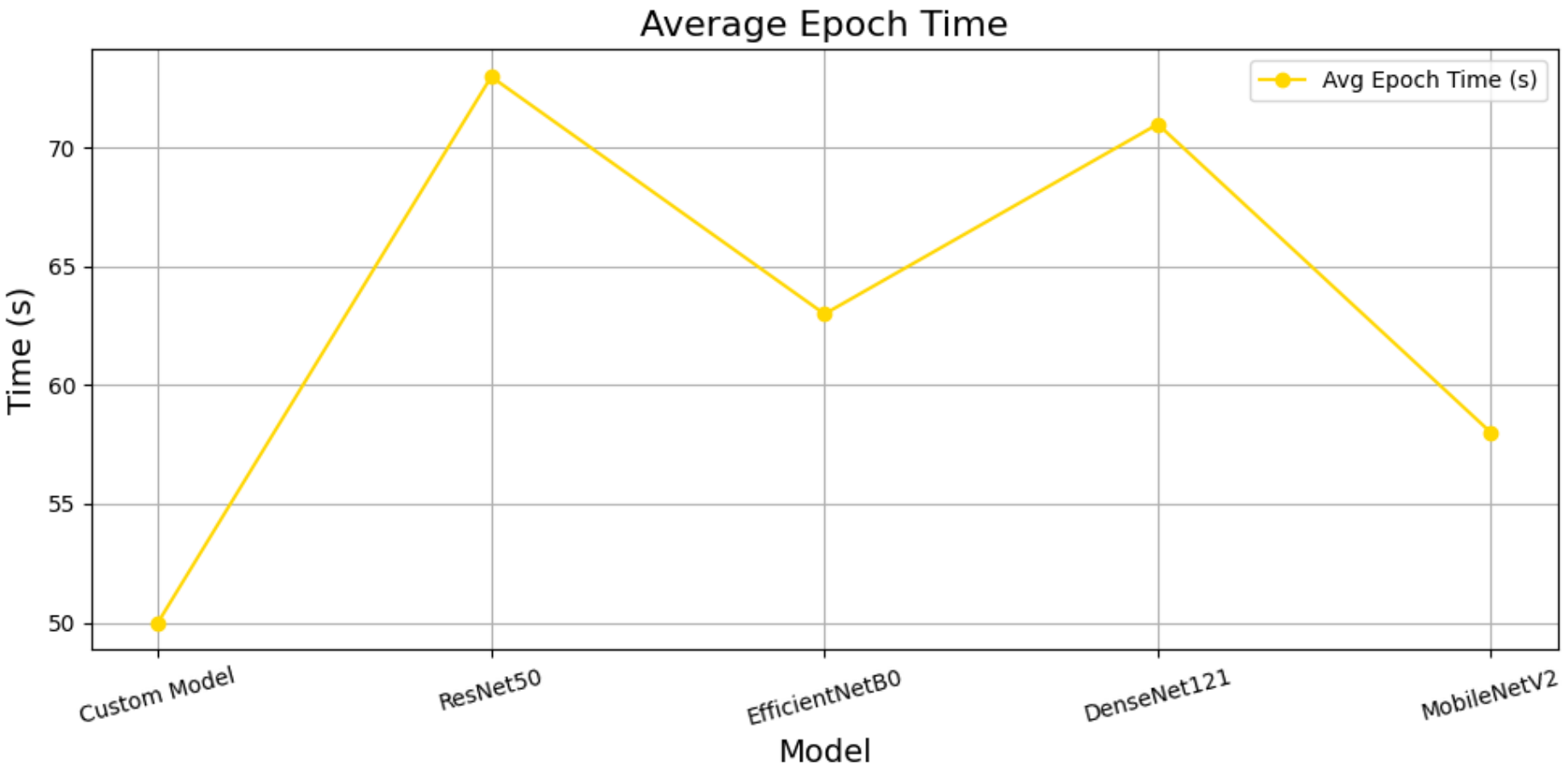

Validation Accuracy
Validation Accuracy
1.000
0.998
0.996
0.994
0.992
0.990
Accuracy
Custom Model
ResNet50
EfficientNetB0
DenseNet121
MobileNetV2
Model

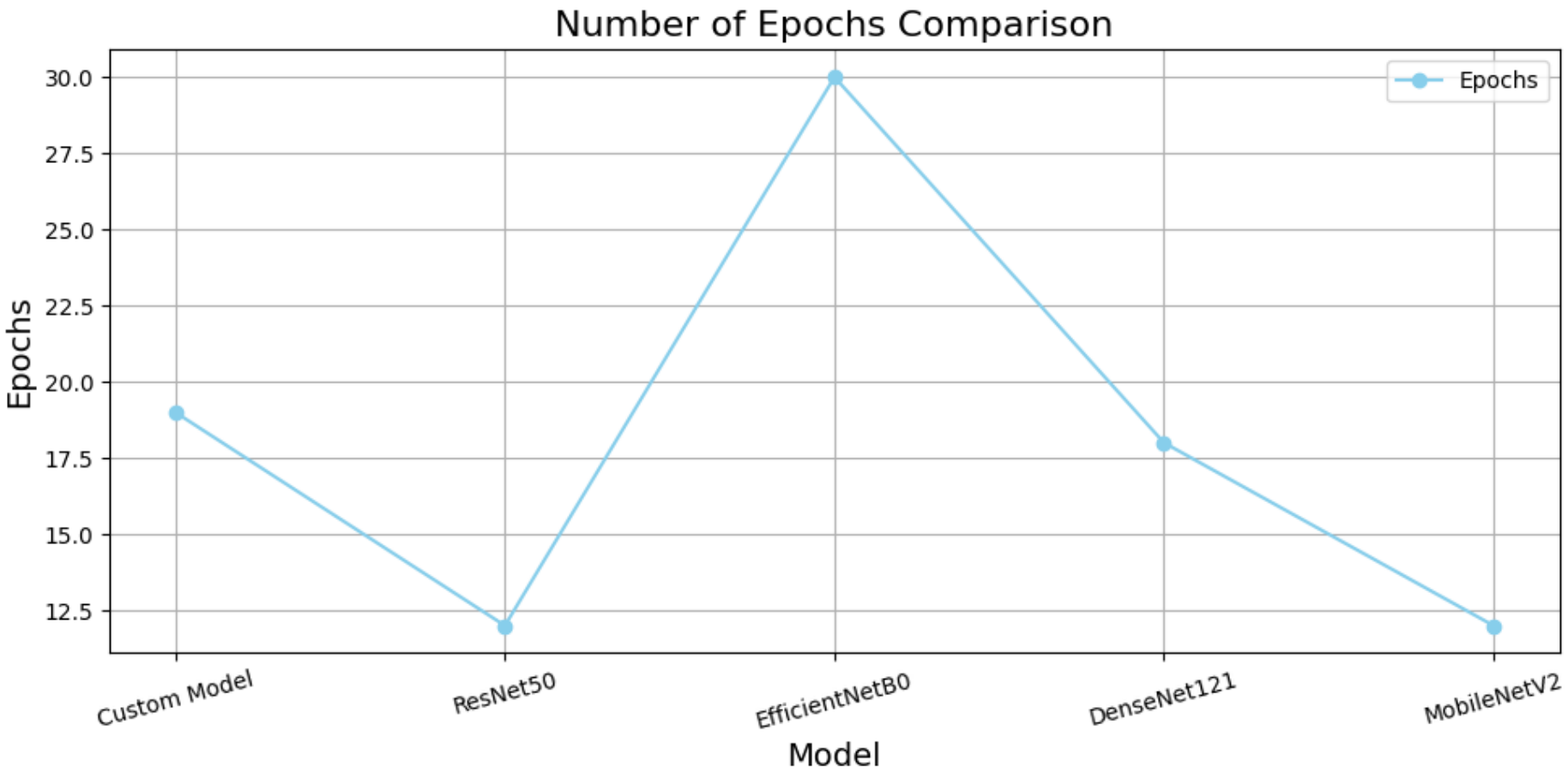

False Positives (FP) and False Negatives (FN) Across Models
False Positives (FP)
False Negatives (FN)
Count
0
2
4
6
8
10
12
Custom Model
ResNet50
EfficientNetB0
DenseNet121
MobileNetV2
Model


False Positive Rate (FPR) and False Negative Rate (FNR) Across Models
False Positive Rate (FPR)
False Negative Rate (FNR)
Rate
0.0035
0.0030
0.0025
0.0020
0.0015
0.0010
0.0005
Custom Model
ResNet50
EfficientNetB0
DenseNet121
MobileNetV2
Model


Validation Loss
Validation Loss
0.016
0.014
0.012
0.010
0.008
0.006
0.004
Loss
Custom Model
ResNet50
EfficientNetB0
DenseNet121
MobileNetV2
Model


ROC AUC
ROC AUC
1.0000
0.9998
0.9996
0.9994
0.9992
0.9990
AUC
Custom Model
ResNet50
EfficientNetB0
DenseNet121
MobileNetV2
Model


Average Epoch Time
Avg Epoch Time (s)
Time (s)
70
65
60
55
50
Custom Model
ResNet50
EfficientNetB0
DenseNet121
MobileNetV2
Model


Number of Epochs Comparison
Epochs
30.0
27.5
25.0
22.5
20.0
17.5
15.0
12.5
Epochs
Custom Model
ResNet50
EfficientNetB0
DenseNet121
MobileNetV2
Model


Total Training Time Comparison
Total Time (s)
1800
1600
1400
1200
1000
800
Custom Model
ResNet50
EfficientNetB0
DenseNet121
MobileNetV2
Model

7. Retrain and Predict

- Final model: ResNet50
- Retrain: using the entire training dataset.
- Apply the best hyperparameters from the pre-trained model: input size, batch size, epoch
- Make final prediction

8. Lesson Learned

- Proper normalization and dataset preparation significantly impacted model performance.
- Adjusting learning rates, batch sizes, and epochs, along with callbacks like EarlyStopping, improved results.
- Pre-trained models like ResNet50 significantly reduced training time and improved accuracy compared to custom models.
- Combining metrics like Accuracy, Validation Loss, and Confusion Matrix ensured a comprehensive understanding of model performance.