Project 1:
Face Recognition

Dicky Adi Naufal Farhansyah | Harianto Adriprasetyo | Ilham Setya Budi
Randy Franstiarajah | Sofia Hana | Yudha Kuswandi

Dicky Adi Group

16 Juli 2023

Introduction

Large-scale CelebFaces Attributes (CelebA) Dataset

- 10,177 number of identities
- 202,599 number of face images
- 5 landmark locations per image
- 40 binary attributes annotations per image

CelebAMask-HQ Dataset

CelebA-Spoof Dataset

CelebA-Dialog Dataset

Annotations

Landmark

lefteye_x, lefteye_y
righteye_x, righteye_y
nose_x, nose_y
leftmouth_x, leftmouth_y
rightmouth_x, rightmouth_y

Bounding Box

x_1
y_1
width
height

Attributes

5_o_Clock_Shadow Arched_Eyebrows Attractive Bags_Under_Eyes Bald Bangs Big_Lips Big_Nose Black_Hair Blond_Hair Blurry Brown_Hair Bushy_Eyebrows Chubby Double_Chin Eyeglasses Goatee Gray_Hair Heavy_Makeup High_Cheekbones Male Mouth_Slightly_Open Mustache Narrow_Eyes No_Beard Oval_Face Pale_Skin Pointy_Nose Receding_Hairline Rosy_Cheeks Sideburns Smiling Straight_Hair Wavy_Hair Wearing_Earrings Wearing_Hat Wearing_Lipstick Wearing_Necklace Wearing_Necktie Young

Target

Dari 40 atribut, kita hanya menggunakan 2, yaitu "**ID**" (image name) & "**Male**"

output_list_attrib

id	Male
000051.jpg	1
000052.jpg	1
000065.jpg	1
000166.jpg	1
000198.jpg	0
000201.jpg	0
000240.jpg	0
000282.jpg	1
000352.jpg	1

Experiments

Perbandingan VGG, ResNet, GoogleNet
Oleh: Ilham Setya Budi

Loss Function: Cross Entropy

VGG
VGG pre-trained
ResNet
ResNet pre-trained
GoogleNet
GoogleNet pre-trained

~~**Multilabel Image Recognition**~~
~~**Oleh: Harianto Adiprasetyo**~~

~~Loss Function: MultiLabelSoftMarginLoss~~

ResNet-18 (Pre-trained)
Predict 39 Attributes (except "Man")

Pytorch Lightning
Oleh: Dicky Adi Naufal Farhansyah

Loss Function: Binary Crossentropy

VGG-16
ResNet-50

Inference Time
Tensorboard

```
'5_o_Clock_Shadow',
'Arched_Eyebrows',
'Attractive',
'Bags_Under_Eyes',
'Bald',
'Bangs',
'Big_Lips',
'Big_Nose',
'Black_Hair',
'Blond_Hair',
'Blurry',
'Brown_Hair',
'Bushy_Eyebrows',
'Chubby',
'Double_Chin',
'Eyeglasses',
'Goatee',
'Gray_Hair',
'Heavy_Makeup',
'High_Cheekbones',
'Male',
'Mouth_Slightly_Open',
'Mustache',
'Narrow_Eyes',
'No_Beard',
'Oval_Face',
'Pale_Skin',
'Pointy_Nose',
'Receding_Hairline',
'Rosy_Cheeks',
'Sideburns',
'Smiling',
'Straight_Hair',
'Wavy_Hair',
'Wearing_Earrings',
'Wearing_Hat',
'Wearing_Lipstick',
'Wearing_Necklace',
'Wearing_Necktie',
'Young'
```

Issues

(with multilabel classification)

Learning Time

Model machine learning harus ada 2:

- Model klasifikasi input image, output multilabel
- Model klasifikasi input tabular, output binary (Male/Female)

Model size lebih besar, convergence lebih lama, learning 2x.
Plus, butuh custom dataset, custom loss function. (no custom dataloader)

CPU vs Cuda

Code yang sudah berjalan dengan CPU, apabila diubah ke Cuda, butuh adjustment dari sisi code juga. Ada data type tensor yang khusus untuk Cuda, sehingga code harus diubah lagi.

Asumsi
(don't assume with data...)

Asumsi logika / nalar manusia lebih baik / dapat membantu performance machine learning. Berdasarkan pengalaman, justru sebaliknya yang terjadi...

Dan, inference time pasti lebih buruk

Performance / Akurasi

Akurasi berkurang. Klasifikasi multilabel justru berpotensi mengurangi input yang dapat kita gunakan untuk klasifikasi.

(konversi ke data multilabel justru menghilangkan data 224 x 224 pixel x 3 RGB matrix, menggantikannya dengan data 39 kolom saja)

Perbandingan VGG, ResNet, GoogleNet
Oleh: Ilham Setya Budi

parameter

Loss Function : CrossEntropyLoss
Optimizer : Adam Optimizer
batch_size = 128
learning_rate = 0.001
epochs = 10

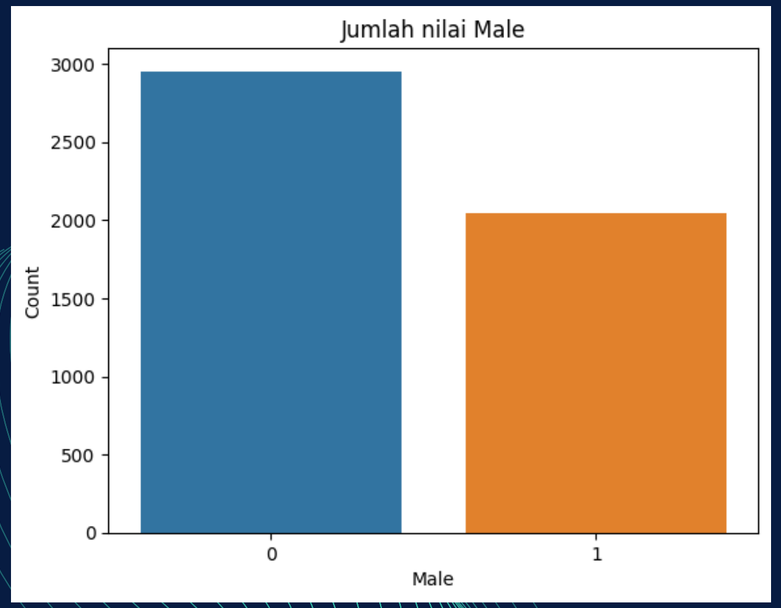
Model

- VGG16
- VGG16 pre-trained
- ResNet18
- ResNet18 pre-trained
- GoogleNet Inception-v1
- GoogleNet Inception-v1 pre-trained

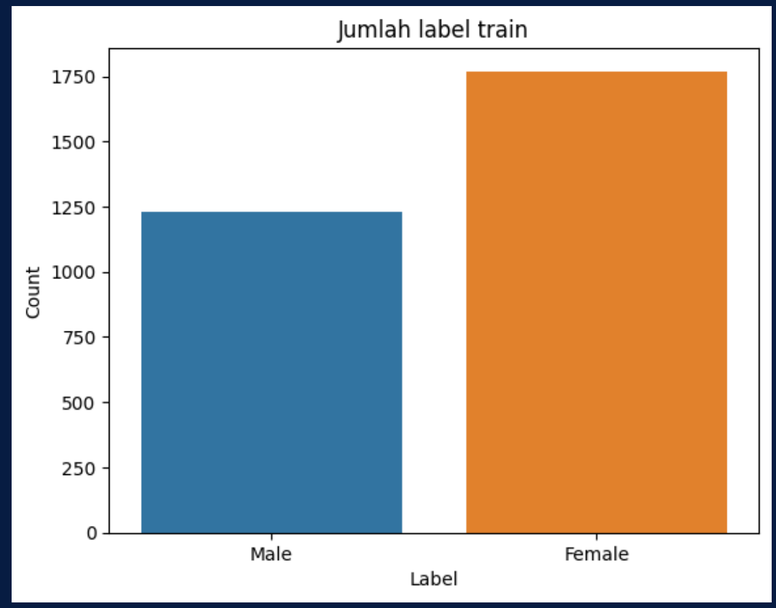
IMAGE TRANSFORMS

```
# Define the transformation
train_transform = transforms.Compose([
    transforms.Resize(256),
    transforms.RandomResizedCrop(224),
    transforms.RandomHorizontalFlip(),
    transforms.ToTensor(),
    transforms.Normalize([0.485, 0.456, 0.406], [0.229, 0.224, 0.225])
])

# Define the validation transformation
val_transform = transforms.Compose([
    transforms.Resize(256),
    transforms.CenterCrop(224),
    transforms.ToTensor(),
    transforms.Normalize([0.485, 0.456, 0.406], [0.229, 0.224, 0.225])
])
```

```python
# Pisahkan fitur (X) dan label (y)
X = dfGenderFix.drop(['imageId', 'Male'], axis=1)
y = dfGenderFix['Male']

# Bagi data menjadi set pelatihan dan uji
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.4, random_state=42)

# Tambahkan kolom baru untuk menandai set pelatihan atau uji
dfGenderFix['Set'] = np.nan
dfGenderFix.loc[X_train.index, 'Set'] = 1
dfGenderFix.loc[X_test.index, 'Set'] = 0

# Konversi tipe data kolom 'Set' menjadi integer
dfGenderFix['Set'] = dfGenderFix['Set'].astype(int)
```



DATA LENGKAP
SEBELUM DI SPLIT

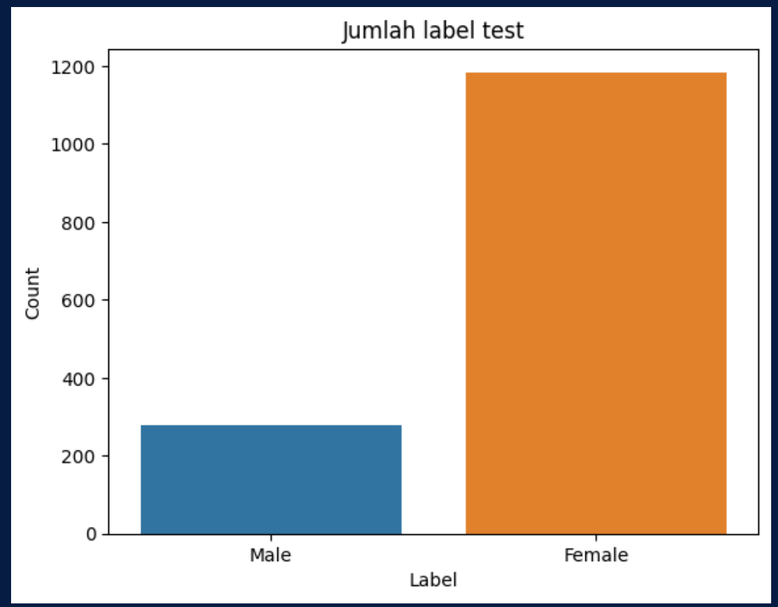


DATA TRAINING

```python
# Mengidentifikasi indeks data yang memenuhi kondisi
indices_to_update = dfGenderFix[(dfGenderFix['Set'] == 0) & (dfGenderFix['Male'] == 1)].index[:538]

# Memperbarui nilai kolom 'Set' menjadi 1 untuk indeks yang diidentifikasi
dfGenderFix.loc[indices_to_update, 'Set'] = 1

# Menghitung jumlah nilai 1 di kolom 'Set' dan nilai 1 di kolom 'Male'
count_1_1 = len(dfGenderFix[(dfGenderFix['Set'] == 1) & (dfGenderFix['Male'] == 1)])

# Menghitung jumlah nilai 1 di kolom 'Set' dan nilai 0 di kolom 'Male'
count_1_0 = len(dfGenderFix[(dfGenderFix['Set'] == 1) & (dfGenderFix['Male'] == 0)])

print("Jumlah nilai 1 di kolom 'Set' dan nilai 1 di kolom 'Male':", count_1_1)
print("Jumlah nilai 1 di kolom 'Set' dan nilai 0 di kolom 'Male':", count_1_0)
```

```
Jumlah nilai 1 di kolom 'Set' dan nilai 1 di kolom 'Male': 1769
Jumlah nilai 1 di kolom 'Set' dan nilai 0 di kolom 'Male': 1769
```

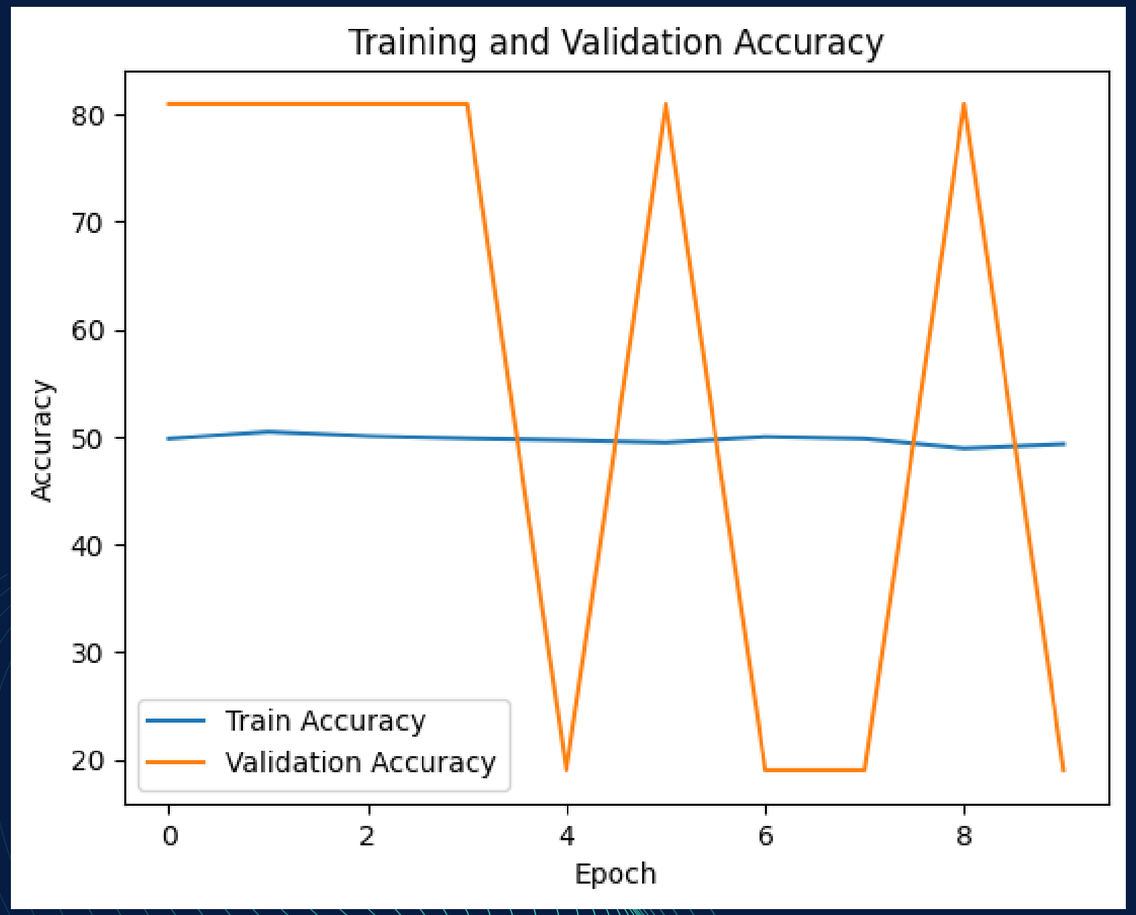
DATA TRAINING
SETELAH PROSES RESAMPLING

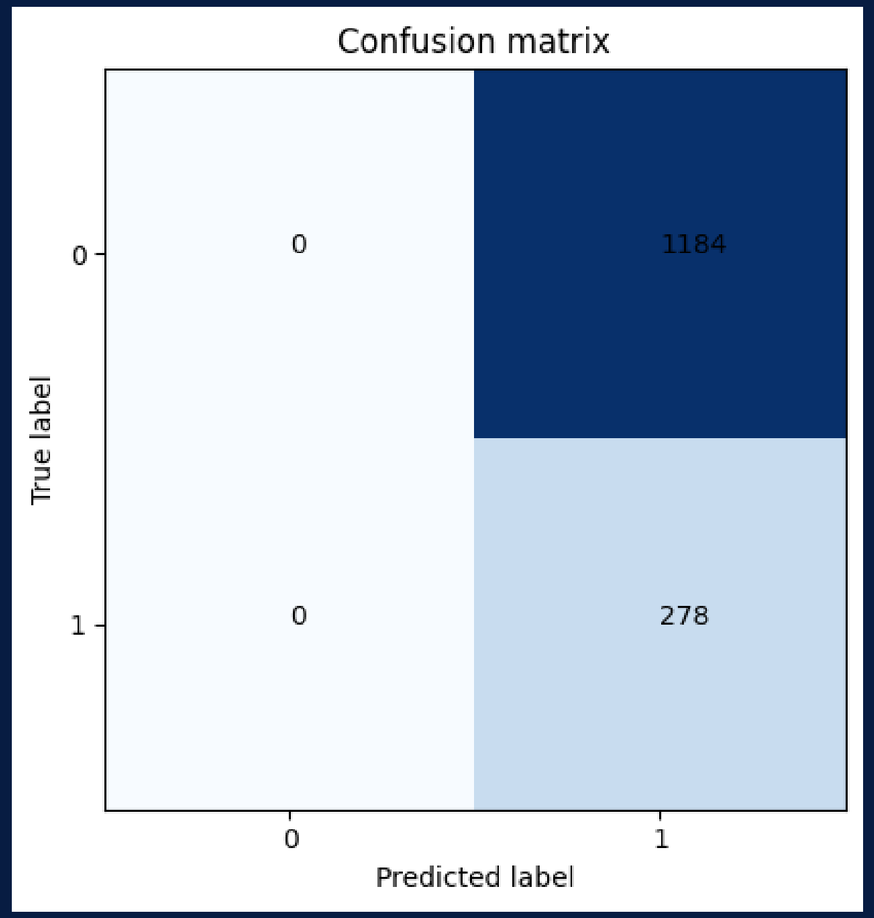


DATA TESTING
SETELAH PROSES RESAMPLING

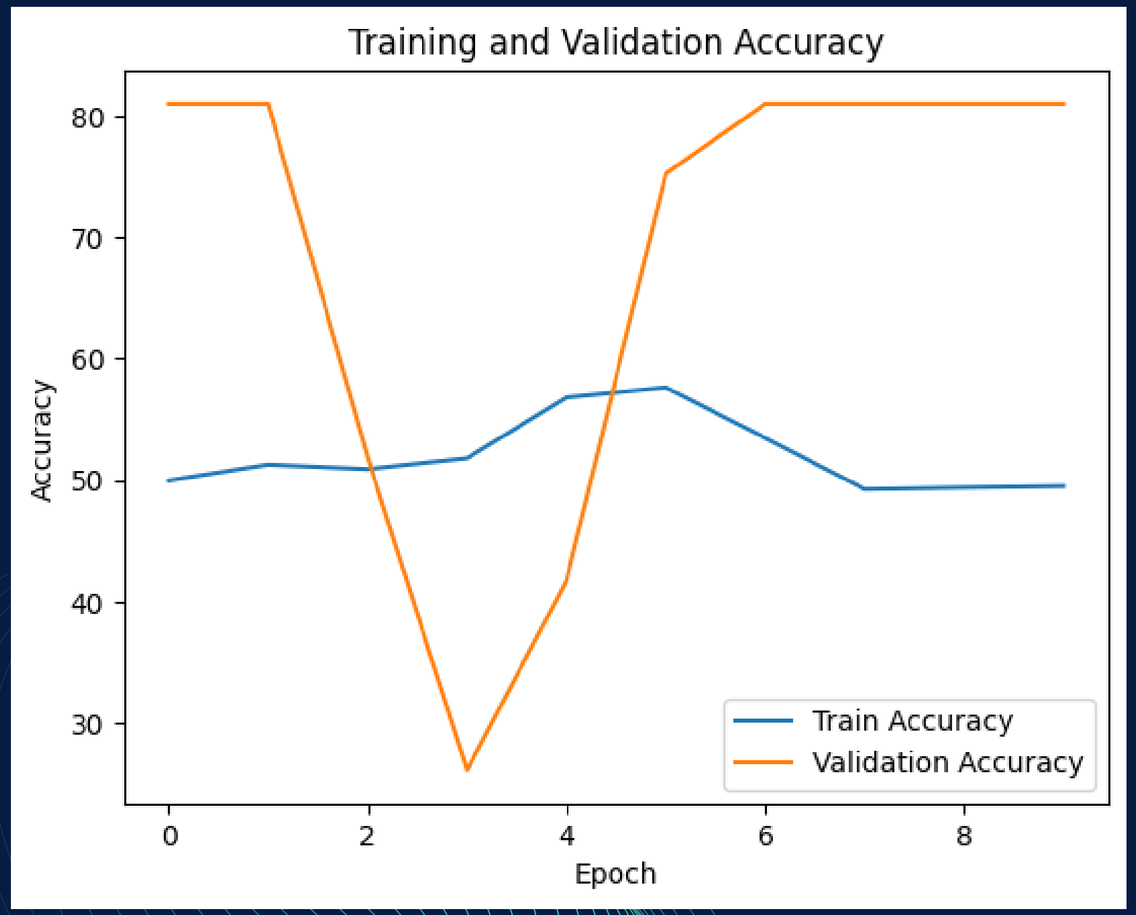
VGG

Accuracy: 278 / 1462
= 19.01%

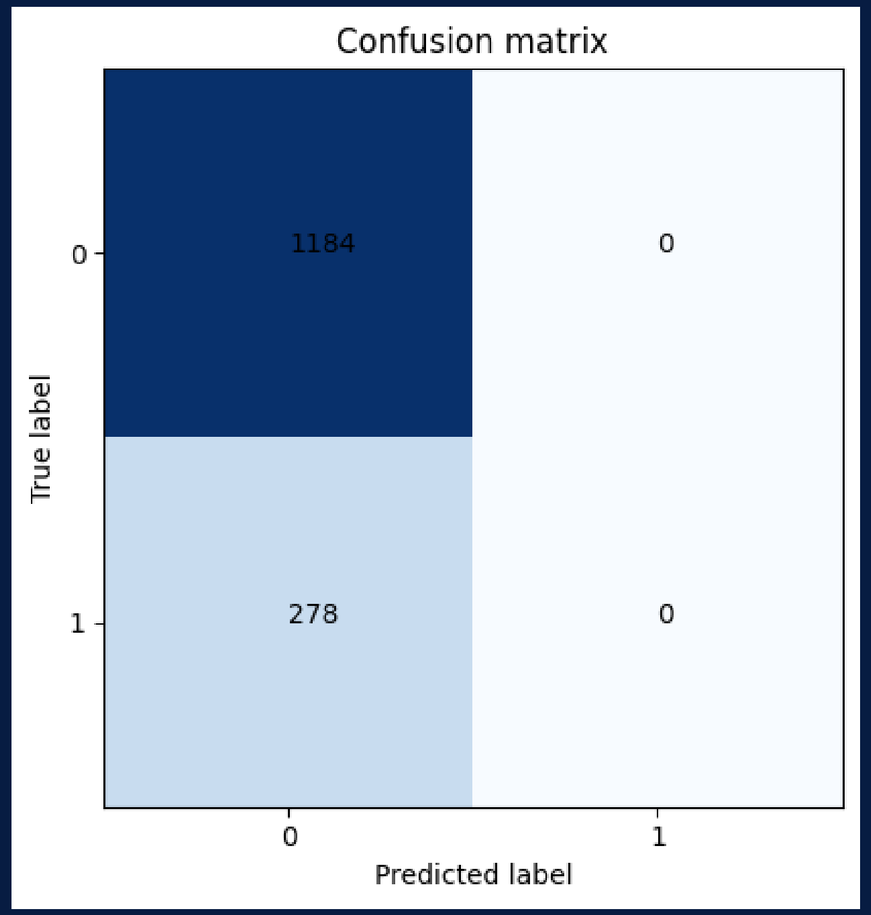




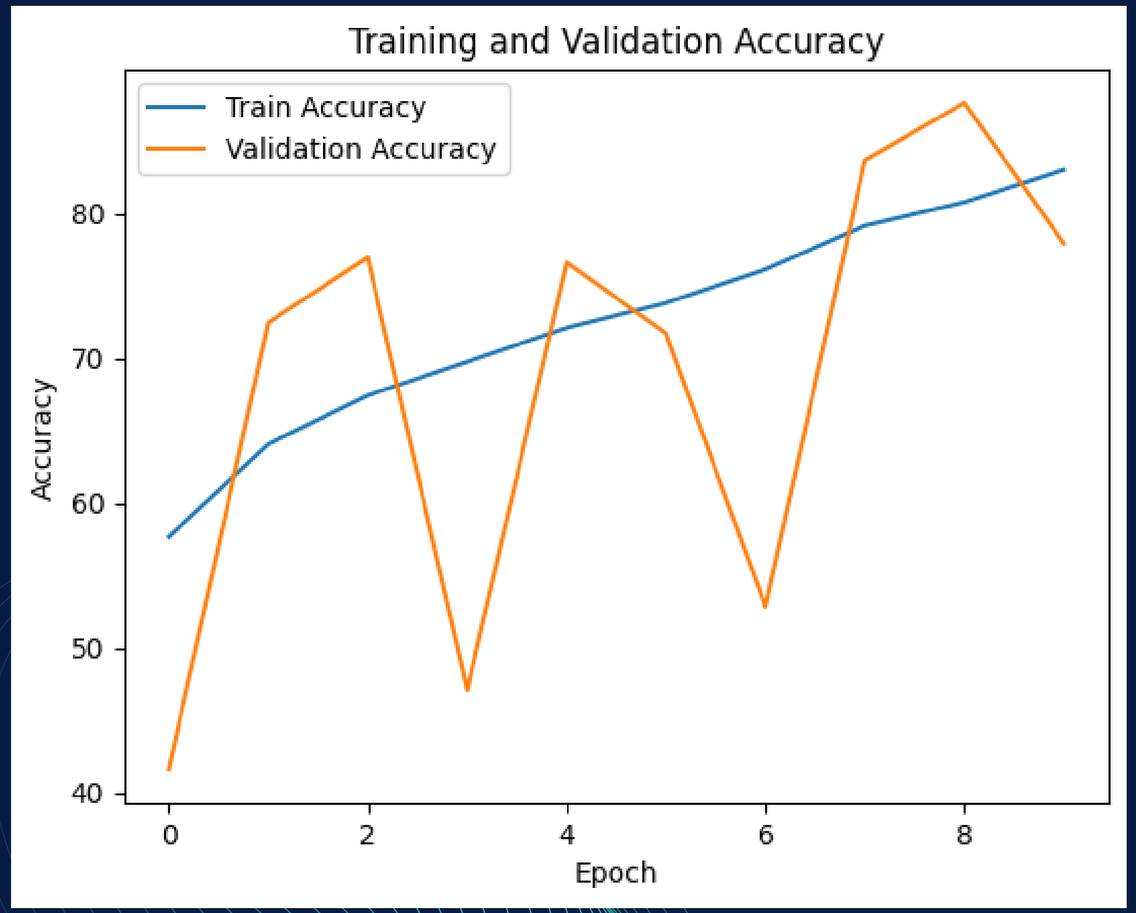
VGG pre-trained



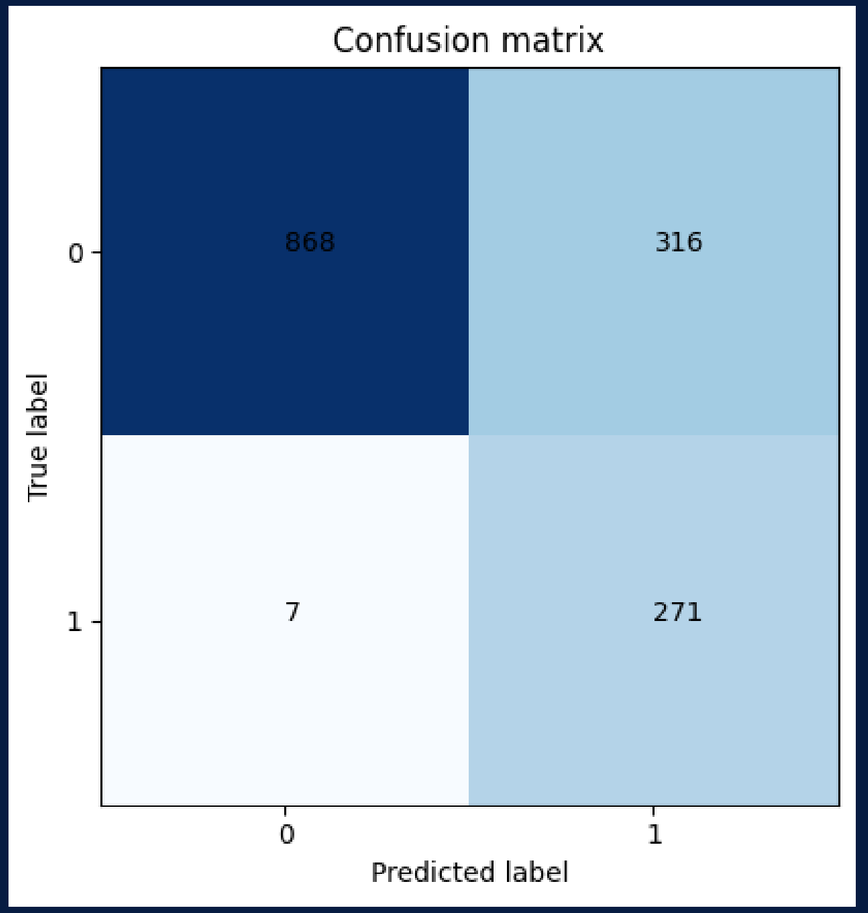
Accuracy: 1184 / 1462 = 80.98%



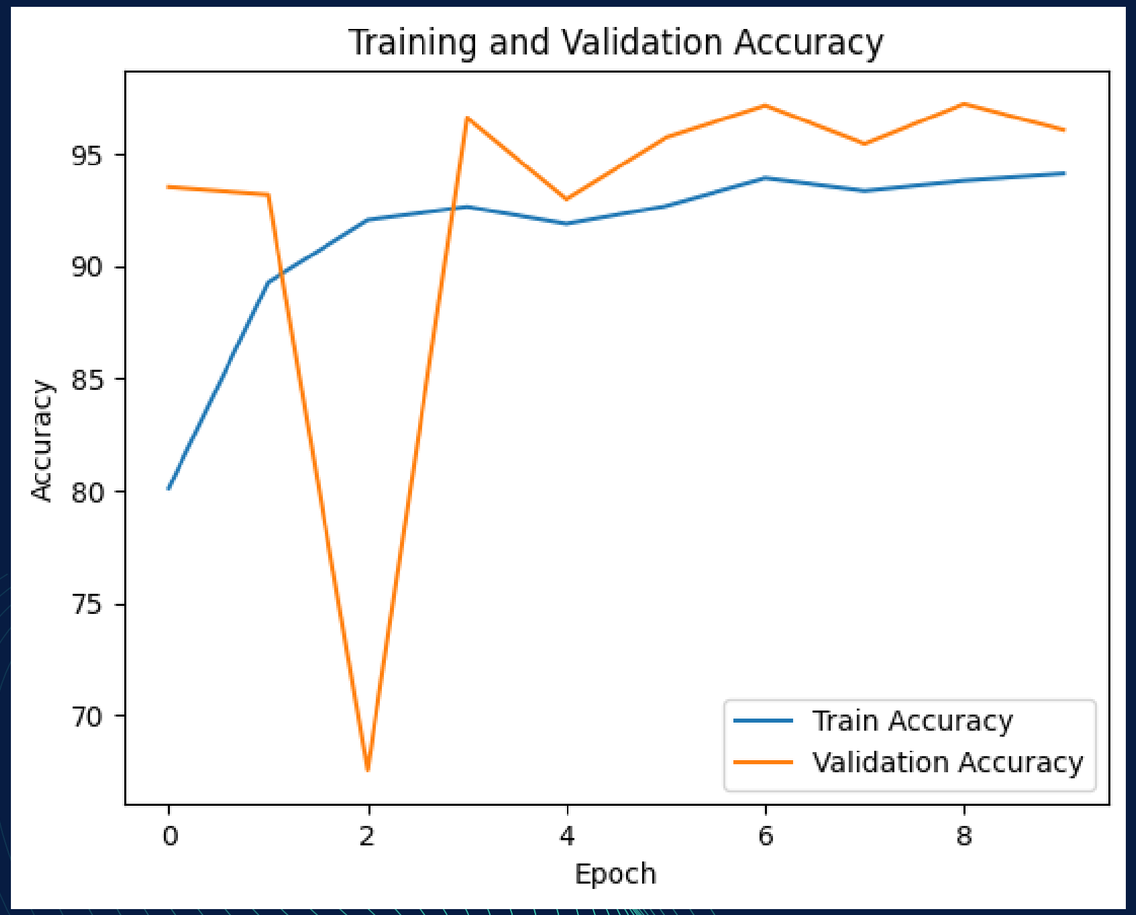
ResNet



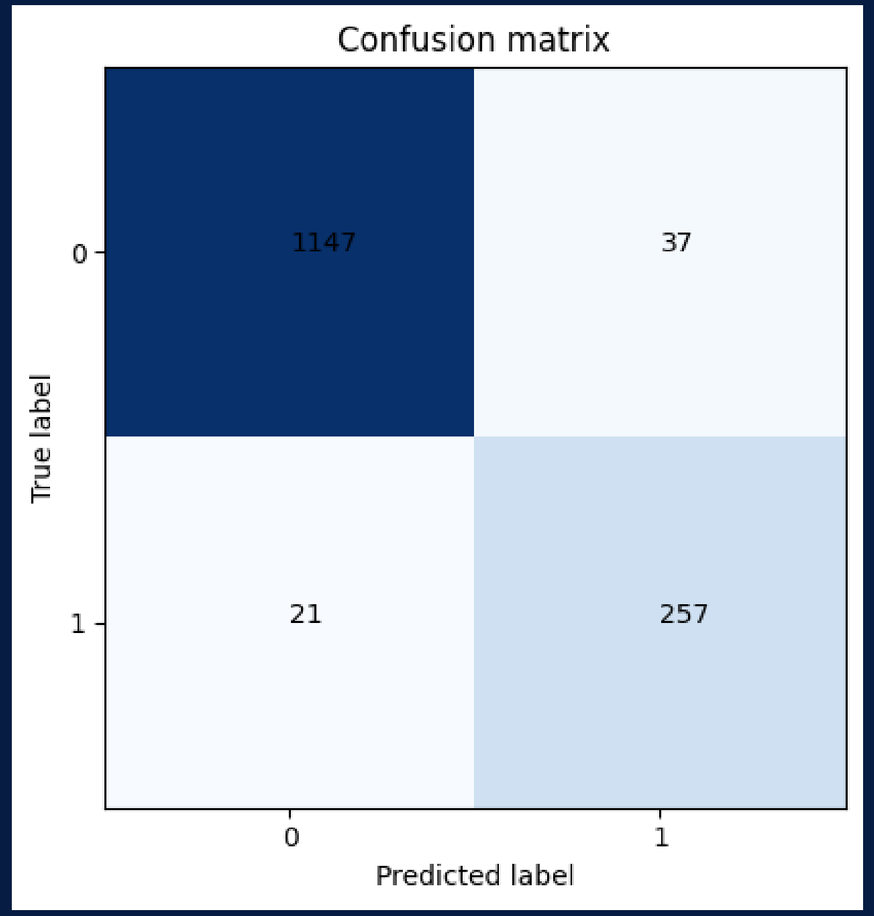
Accuracy: 1139 / 1462 = 77.91%



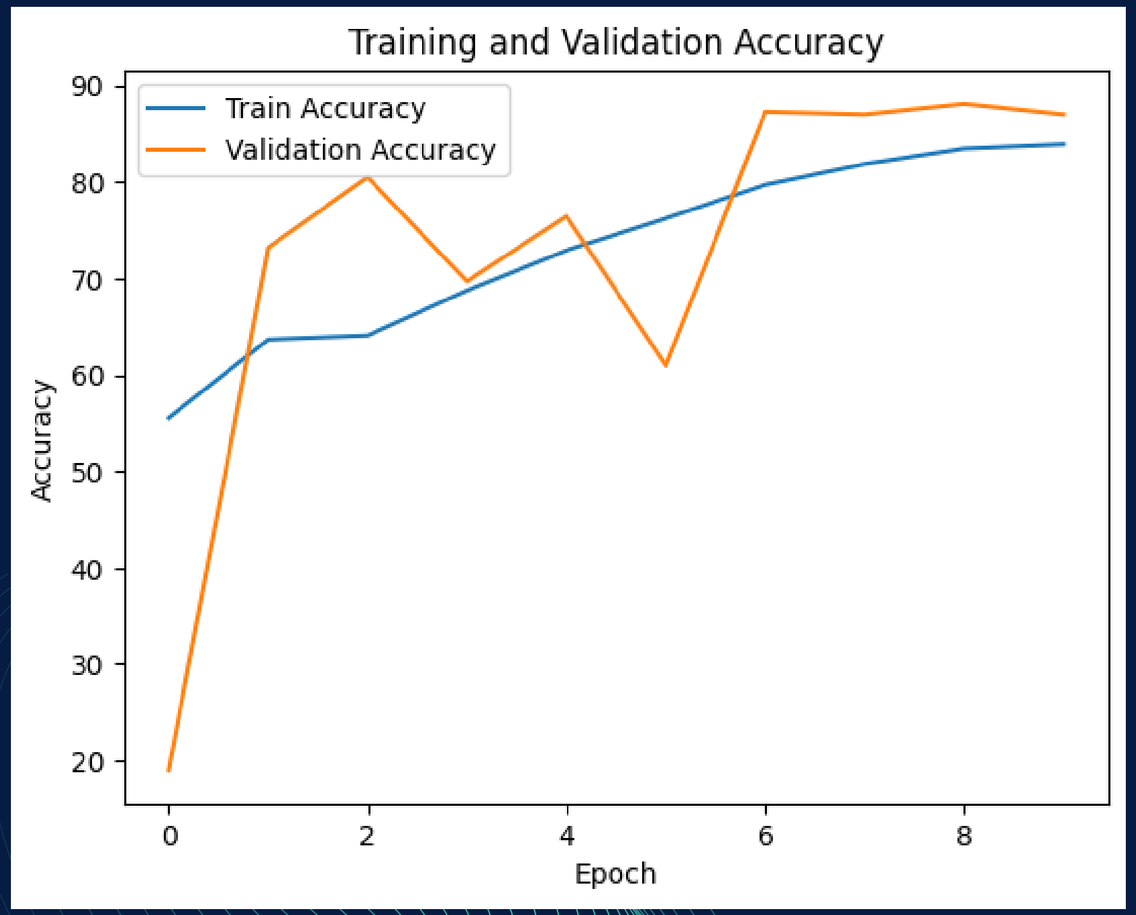
ResNet pre-trained



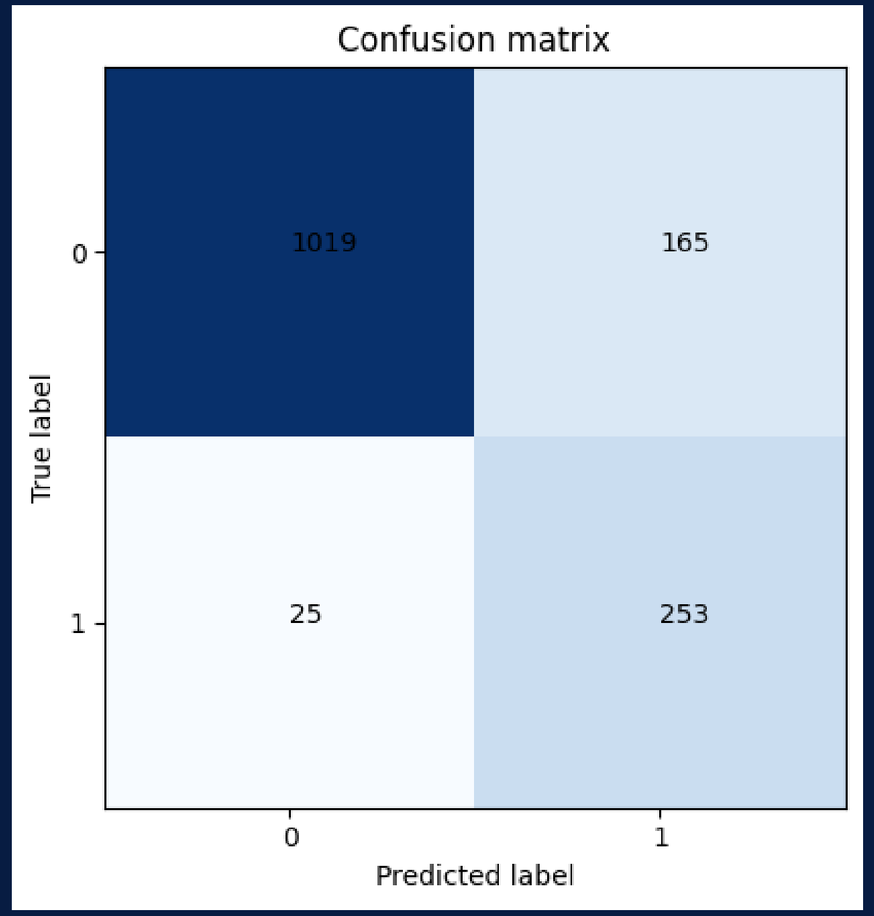
Accuracy: 1404 / 1462 = 96.03%



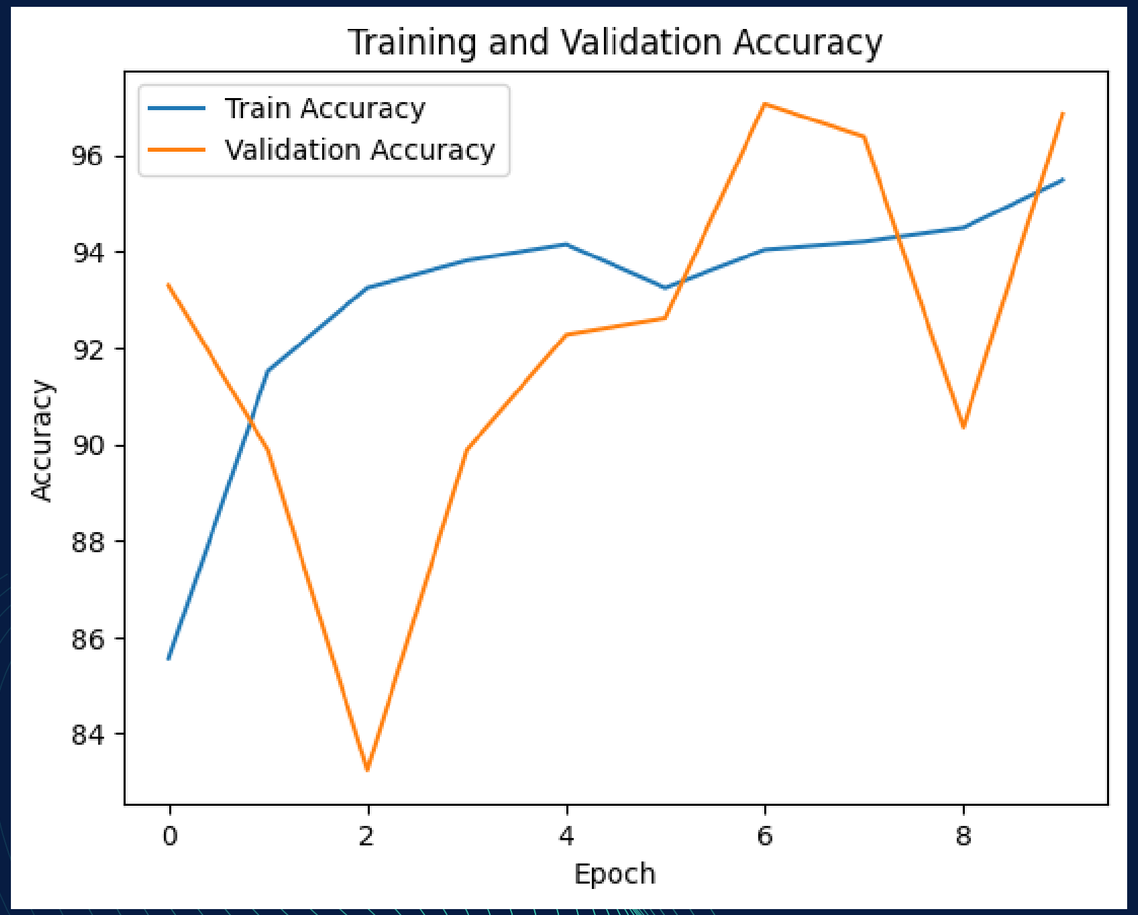
GoogleNet



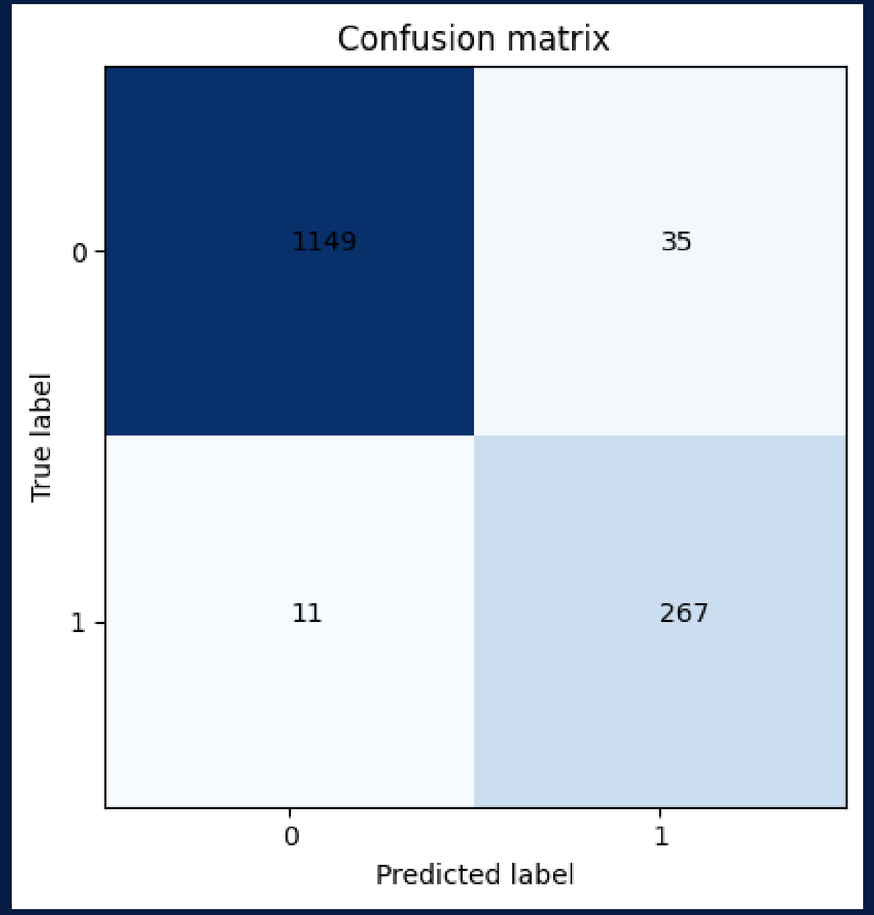
Accuracy: 1272 / 1462 = 87.00%



GoogleNet pre-trained



Accuracy: 1416 / 1462 = 96.85%



GoogleNet pre-trained

Accuracy: 96.85%

ResNet pre-trained

Accuracy: 96.03%

GoogleNet

Accuracy: 87.00%

Results

Loss Function : CrossEntropyLoss

Optimizer : Adam Optimizer

batch_size = 128

learning_rate = 0.001

epochs = 10

Pytorch Lightning
Oleh: Dicky Adi Naufal Farhansyah

PyTorch Lightning is an ML framework that helps AI engineer or data scientist to develop deep learning model without worrying about the messy and complex code in training iterative. Even though it helps to reduce the messy and complex code, as an AI engineer and data scientist we still need to adapt our coding style to fit in PyTorch Lightning code template. PyTorch Lightning also gives some good extra feature that helps ML developer to achieve good performances. This extra feature is used in this project, therefore we will explain this later.

Problems

The main goals of this project is to successfuly predict whether an image of a person is a female or male or can be described as a Gender Classification

Problem 1

Even though the dataset is imbalance, in this experiment we ignore the imbalance dataset

Problem 2

Huge amount of dataset can affect the complexity of the model

Problem 3

Architecture complexity cost huge computation processing

Model and Architecture Description

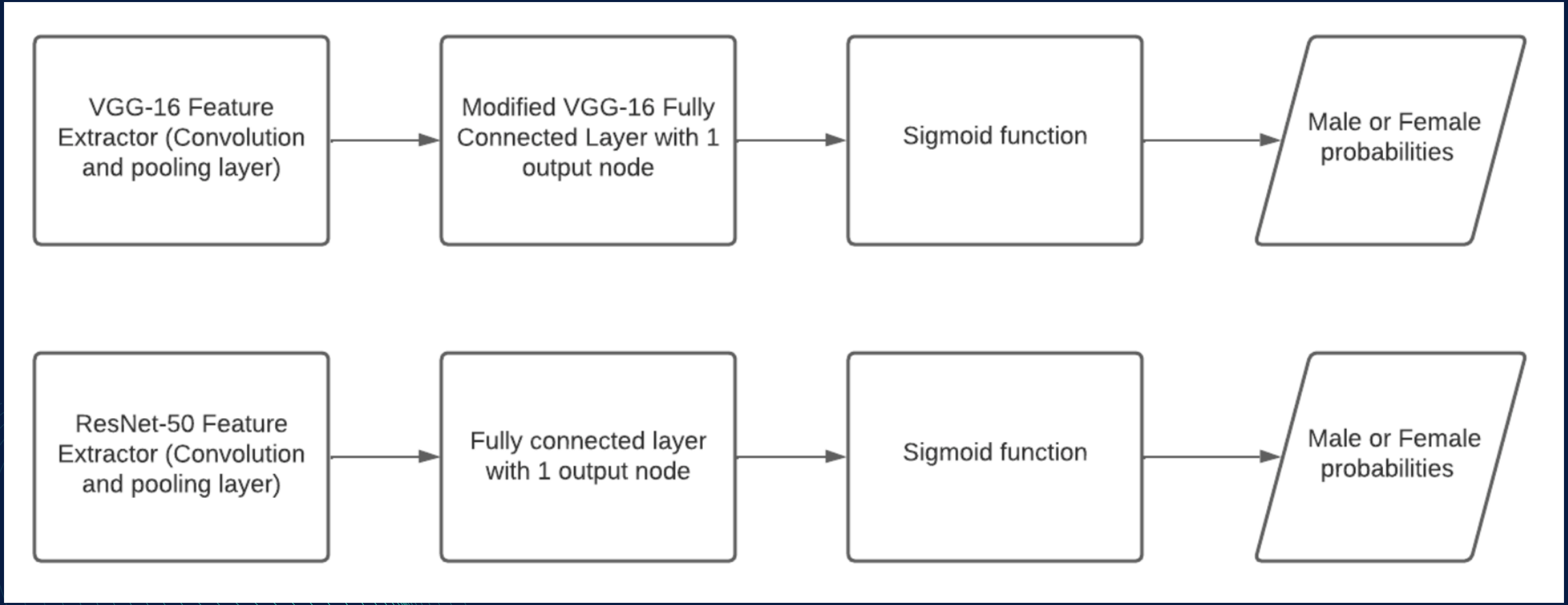
VGG-16 Transfer Learning

- Loss Function : Binary Cross Entropy with Logits Loss
- Threshold : 0.5 (50%) of prediction confidence
- Optimizer : Adam Optimizer
- Learning Rate : To be determined by PyTorch Lightning Learning Rate Finder
- Weight decay (L2 regularizer) : 0.001
- Batch Size : 32

ResNet-50 Transfer Learning

- Loss Function : Binary Cross Entropy with Logits Loss
- Threshold : 0.5 (50%) of prediction confidence
- Optimizer : Adam Optimizer
- Learning Rate : To be determined by PyTorch Lightning Learning Rate Finder
- Weight decay (L2 regularizer) : 0.0001
- Batch Size : 32

Architecture design



PyTorch Lightning Learning rate finder

In deep neural network learning rate is the most important hyper-parameter to tune for training deep neural networks.

A brief overview about this method is the method selecting the learning rate based on cyclical processing where the method set the upper and lower bound of the learning rate and run only a few epoch of training step to find a good learning rate where it has the lowest loss.

Training with cyclical learning rates instead of fixed values achieves improved classification accuracy without a need to tune and often in fewer iterations.This policy is easy to implement and unlike adaptive learning rate methods, incurs essentially no additional computational expense.

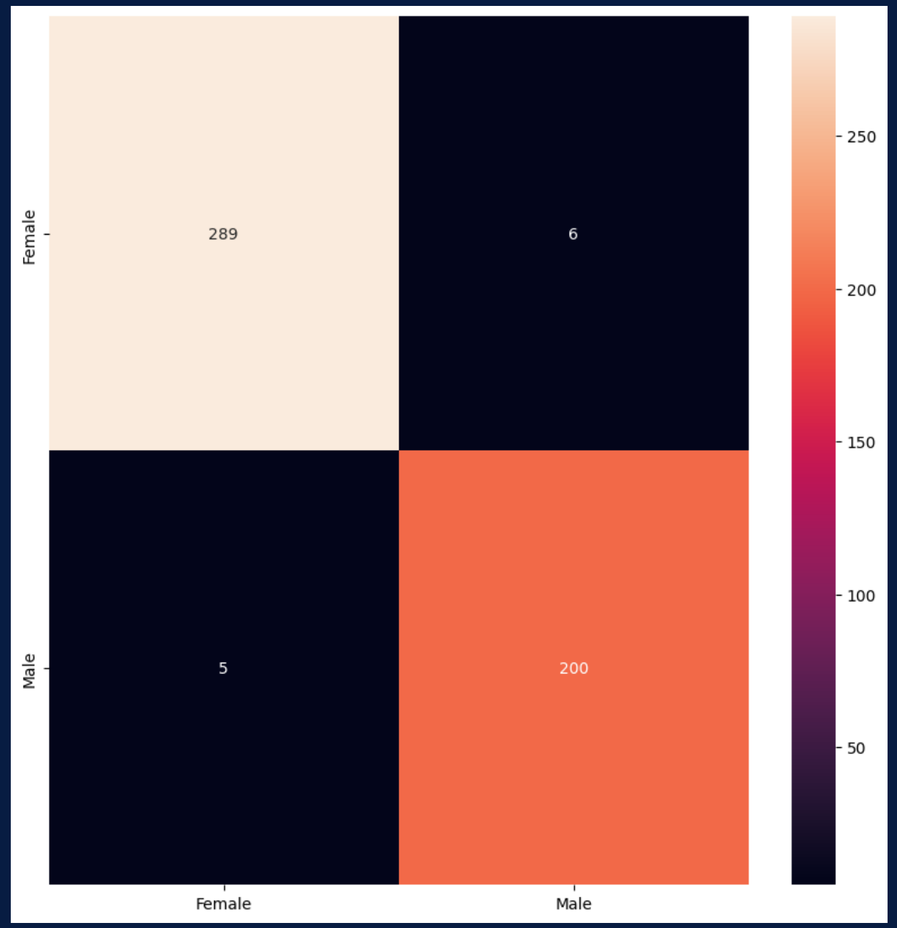
However, this paper has not explored the full range of applications for cyclic learning rate methods.Therefore we need to validate this by reading and understand the research paper of this method which is provided at PyTorch Lightning documentation

Model performances

ResNet-50 Transfer Learning

	precision	recall	f1-score	support
Female	0.98	0.98	0.98	295
Male	0.97	0.98	0.97	205
accuracy			0.98	500
macro avg	0.98	0.98	0.98	500
weighted avg	0.98	0.98	0.98	500

Inference time for 500 images : 1.5192 seconds

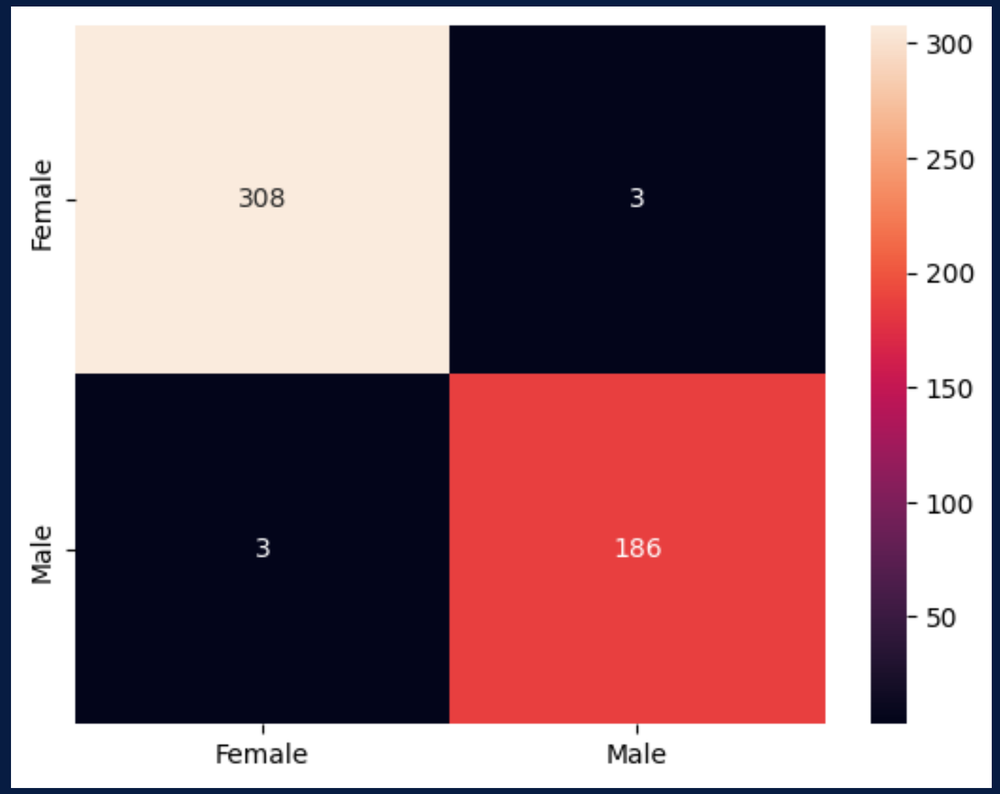


Model performances

VGG-16 Transfer Learning

	precision	recall	f1-score	support
Female	0.99	0.99	0.99	311
Male	0.98	0.98	0.98	189
accuracy			0.99	500
macro avg	0.99	0.99	0.99	500
weighted avg	0.99	0.99	0.99	500

Inference time for 500 images: 2.0529 seconds



Correct prediction

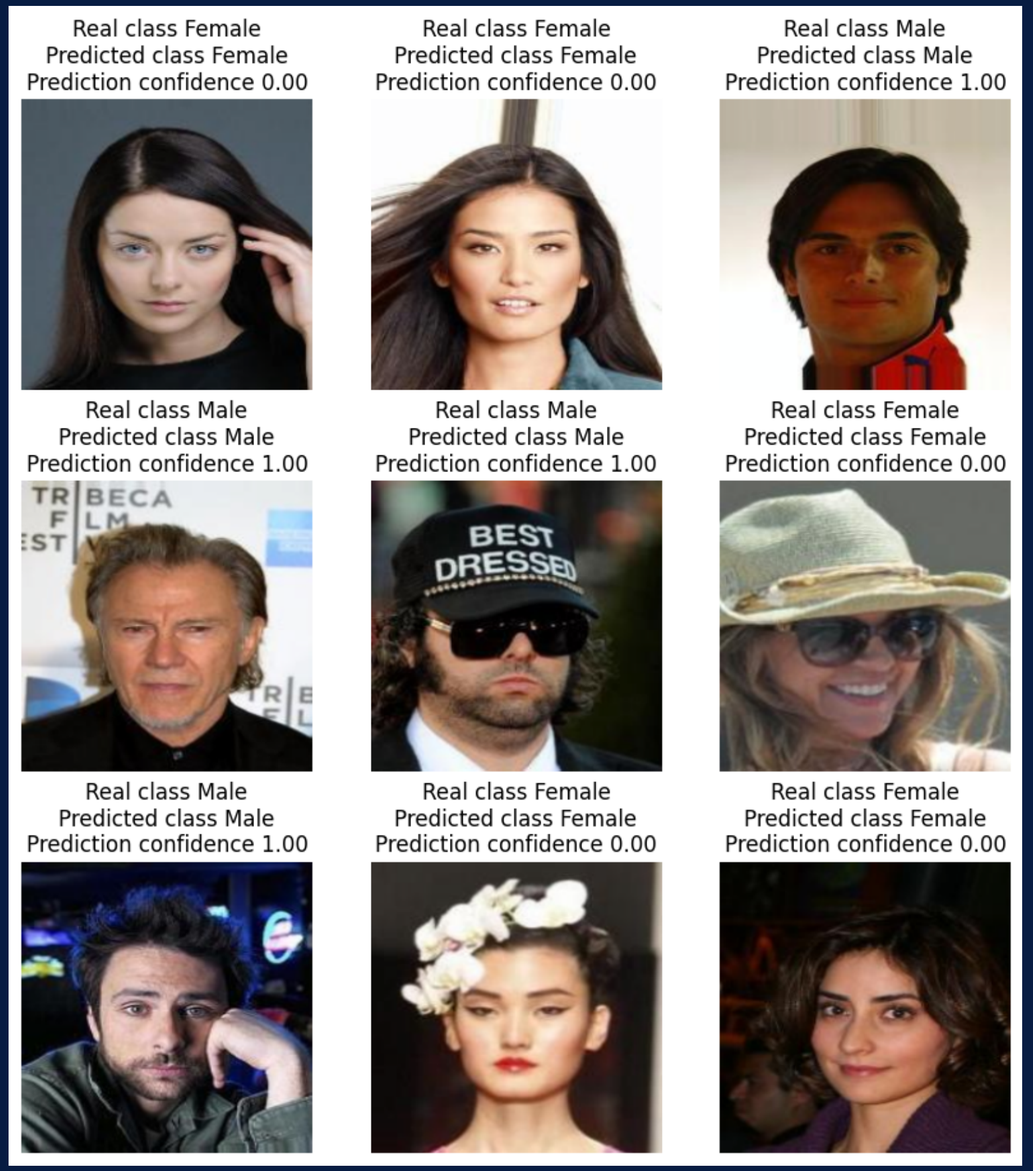
ResNet-50 Transfer Learning

Incorrect prediction

ResNet-50 Transfer Learning

Correct prediction

VGG-16 Transfer Learning

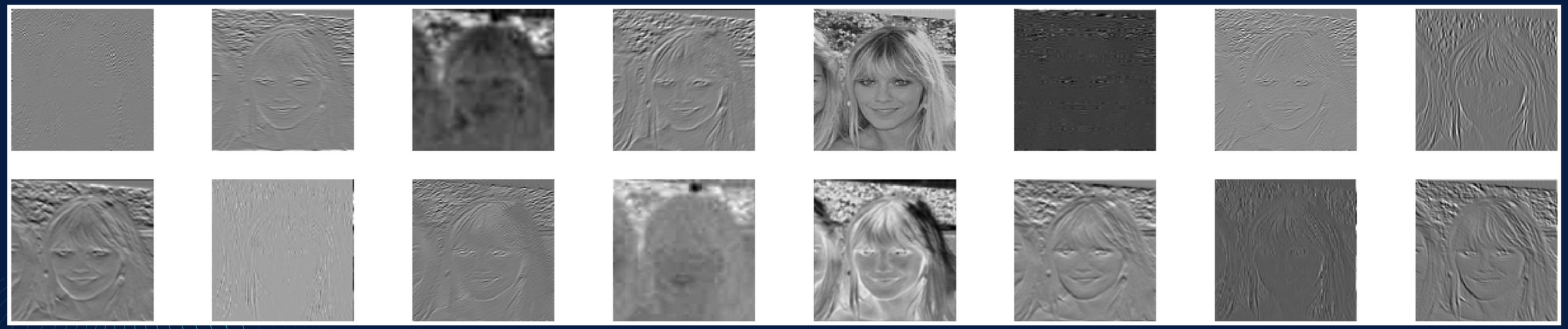


Incorrect prediction

VGG-16 Transfer Learning

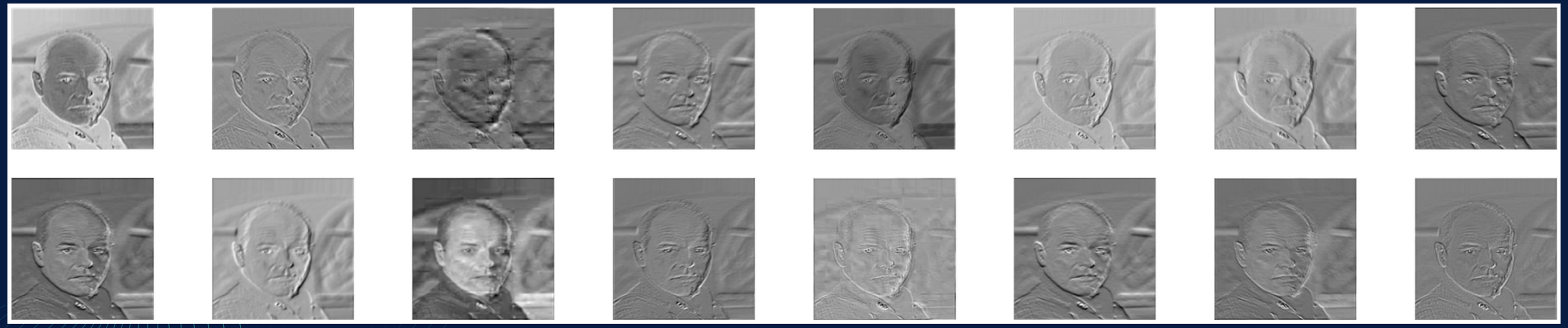
Convolution Layer visualization

ResNet-50 Transfer Learning



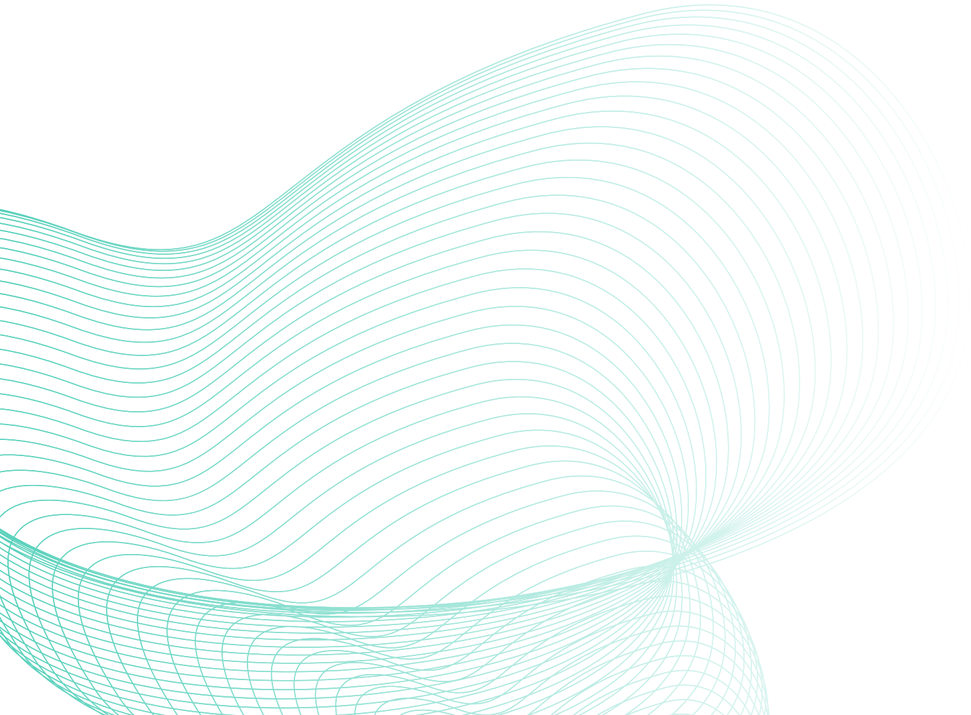
Convolution Layer visualization

VGG-16 Transfer Learning



Conclusion

Based on experiment before, the model successfully predicted the gender of the person within the image even there are some of reasonable incorrect predictions



Solution 1

Based on the incorrect prediction, having an imbalance dataset really affect the model when predicting the gender. Therefore, handling imbalance dataset is a good thing to start before starting the modelling.

Solution 2

When dealing with a huge amount of computational cost. Employing transfer learning along side fine tuning is a good way to start even there are some limitations.

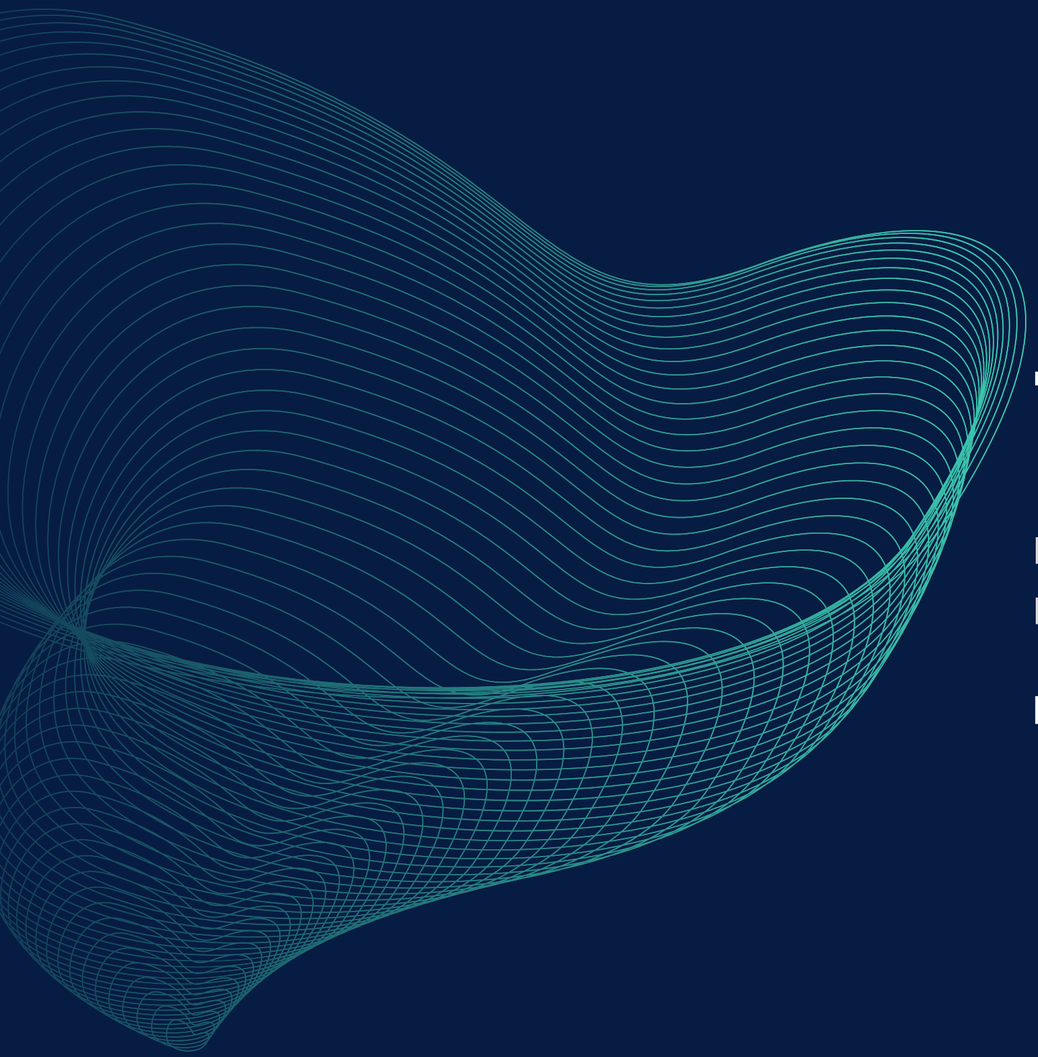
Solution 3

PyTorch Lightning is a useful framework because of the extra feature and relatively simple to code also when it comes to parallel training PyTorch Lightning really helps to handle the small things that you need to handle.

Solution 4

Since we are using pre-trained architecture, it is a good thing to employ regularization to overcome overfitting.

Tensorboard Learning curve



Thank you for your attention!

Dicky Adi Naufal Farhansyah | Harianto Adriprasetyo | Ilham Setya Budi
Randy Franstiarajah | Sofia Hana | Yudha Kuswandi

Dicky Adi Group

16 Juli 2023