Image Classification with CNN models

Machine Learning Final Project

Fall 2023-2024

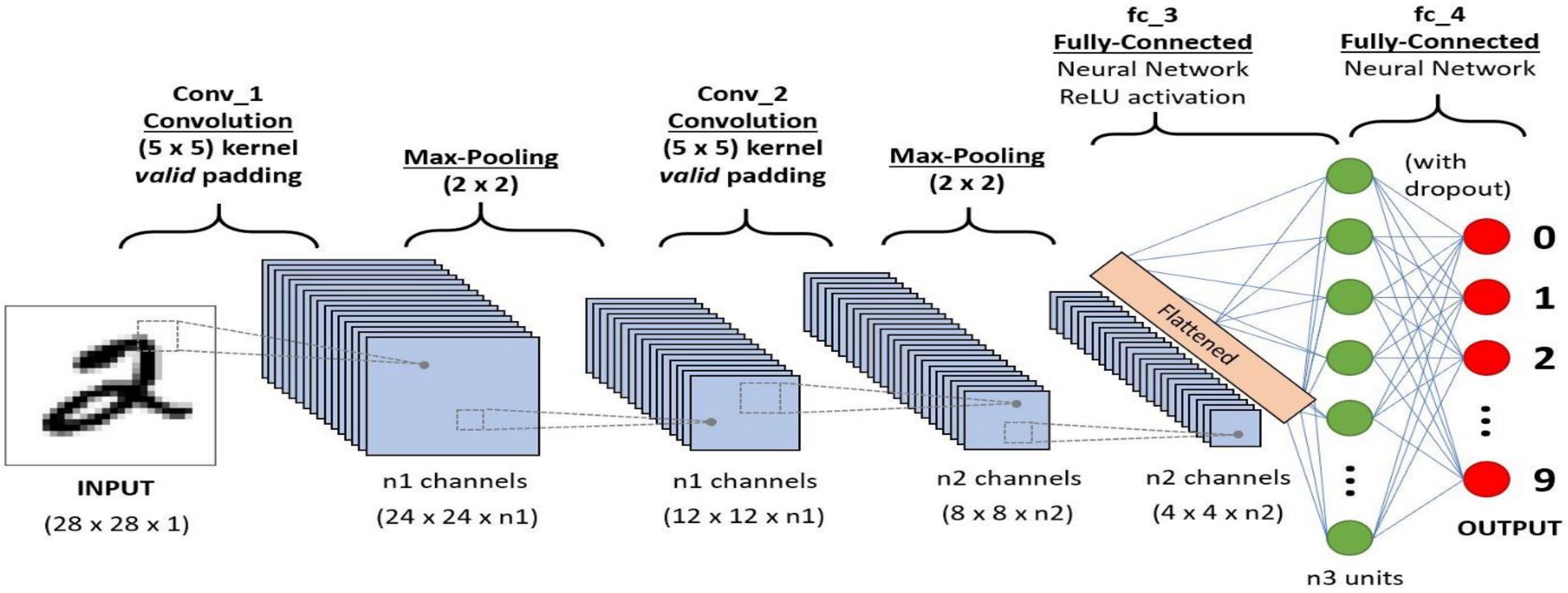
Professor : Dr. Behnam Bahrak

Presenter : Ali Zahedzadeh

mail: alizahedzadeh7@gmail.com

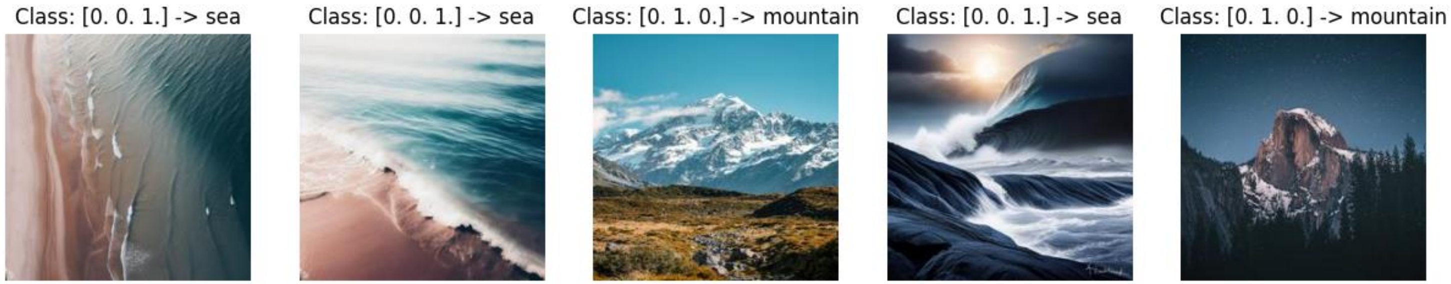
Tehran Institute for Advanced Studies

- Convolutional neural network



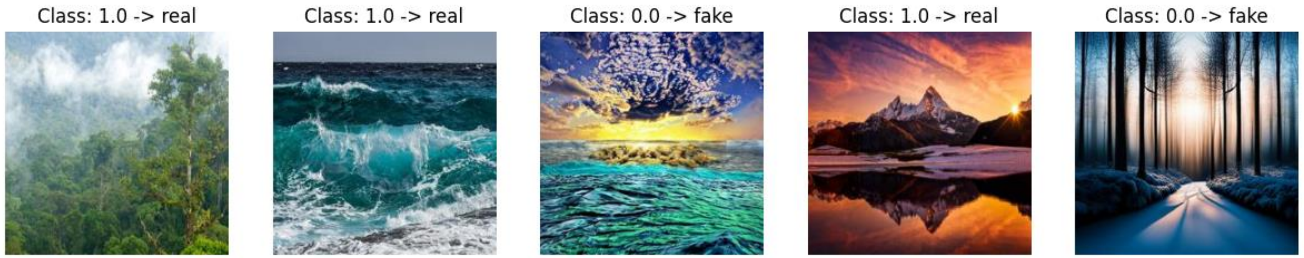
- Task 1

Determine whether the image depicts a sea, mountain, or jungle.



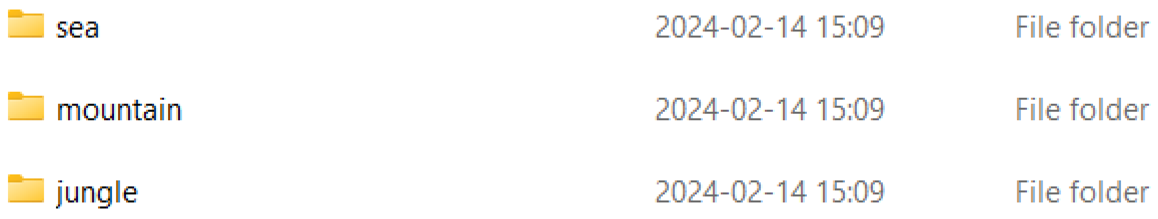
- Task 2

Identify whether the image is real or fake.



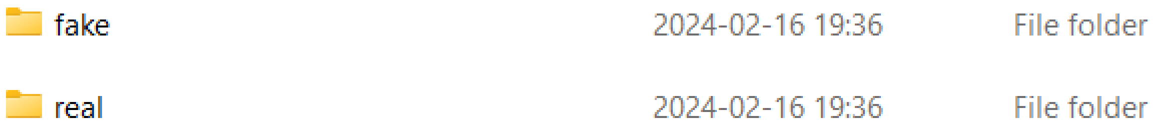
About Dataset

- Sea, Mountain, or Jungle



sea	2024-02-14 15:09	File folder
mountain	2024-02-14 15:09	File folder
jungle	2024-02-14 15:09	File folder

- Real or Fake



fake	2024-02-16 19:36	File folder
real	2024-02-16 19:36	File folder

Make Dataset – Function

```
def Make_Dataset(
    data_path,
    batch_size,
    random_seed,
    class_mode,
    data_augmentation = 'default’):
```

Data Augmentation

- Random Rotation (ImageDataGenerator)

`rotation_range :`

is a value in degrees (0-180), a range within which to randomly rotate pictures

Class: [0. 0. 1.] -> sea

Class: [0. 0. 1.] -> sea

Class: [0. 1. 0.] -> mountain

Class: [0. 0. 1.] -> sea

Class: [0. 1. 0.] -> mountain

Data Augmentation

- Gaussian Blur (ImageDataGenerator)

```
def preprocess_blur(x):
    if np.random.rand() < 0.5:
        return x
    else:
        return cv2.GaussianBlur(x, (15, 15), 0)
```

Class: [0. 0. 1.] -> sea

Class: [0. 0. 1.] -> sea

Class: [0. 1. 0.] -> mountain

Class: [0. 0. 1.] -> sea

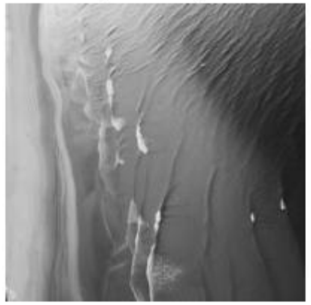
Class: [0. 1. 0.] -> mountain

Data Augmentation

- GrayScale (ImageDataGenerator)

```
def preprocess_grayscale(image):
    # Convert to grayscale
    gray = cv2.cvtColor(image, cv2.COLOR_RGB2GRAY)
    # Add third channel to make it (height, width, 3)
    gray = cv2.cvtColor(gray, cv2.COLOR_GRAY2RGB)
    return gray
```

Class: [0. 0. 1.] -> sea



Class: [0. 0. 1.] -> sea

Class: [0. 1. 0.] -> mountain

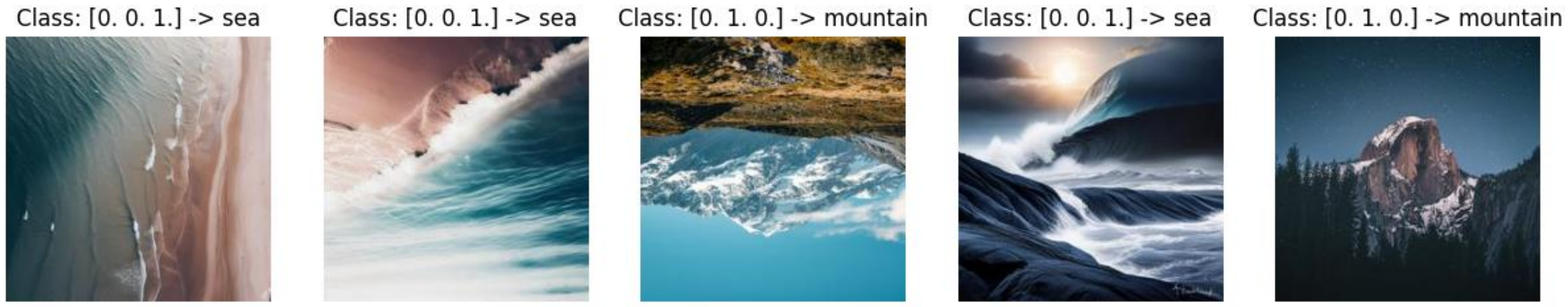
Class: [0. 0. 1.] -> sea

Class: [0. 1. 0.] -> mountain

Data Augmentation

- RandomFlip

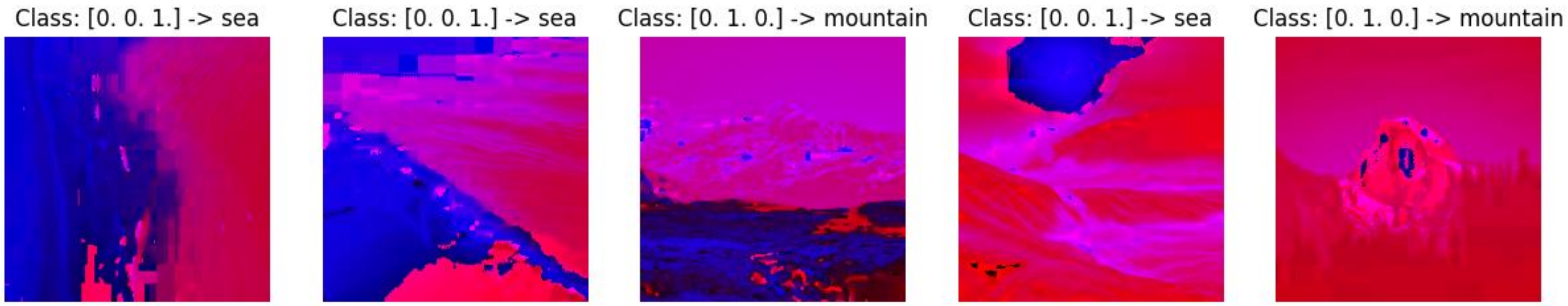
```
horizontal_flip = True
vertical_flip = True
```



Data Augmentation

- Color Jitter

```
def preprocess_colorjitter(image):
    return cv2.cvtColor(image,cv2.COLOR_RGB2HSV)
```



Class: [0. 0. 1.] -> sea

Class: [0. 0. 1.] -> sea

Class: [0. 1. 0.] -> mountain

Class: [0. 0. 1.] -> sea

Class: [0. 1. 0.] -> mountain

Data Augmentation

- MixUp

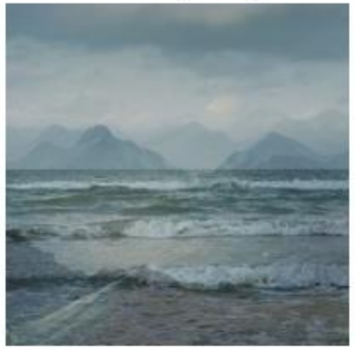
```
class MixupImageDataGenerator()
```

Class: 1.00, 0.00, 0.00

Class: 0.00, 1.00, 0.00

Class: 0.00, 1.00, 0.00

Class: 0.00, 0.24, 0.76



Class: 0.00, 1.00, 0.00

Models

- CNN from Scratch
- ResNet-152
- MobileNetV3

Optimizer

- SGD
- Adam
- AdamW
- RMSprop
- SAM

Make_Models – Function

```python
def Make_Model(
   task,
   model_name,
   optimizer,
   input_shape = (224, 224, 3)):
```

Learning Rate Schedular

- LinearLR

```python
def linear_decay(epoch):
    initial_lr = 0.01
    decay_rate = 0.1
    lr = initial_lr * (1 - decay_rate * epoch)
    return max(lr, 0.0)
```

Learning Rate Schedular

- ExponentialLR

```
def exponential_decay(epoch):
    initial_lr = 0.01
    decay_rate = 0.9
    lr = initial_lr * np.exp(-decay_rate * epoch)
    return max(lr, 0.0)
```

Learning Rate Schedular

- CyclicLR

```
class CyclicLR(Callback):
```

Learning Rate Schedular

- StepLR

```python
def step_decay(epoch):
    initial_lr = 0.01
    drop = 0.5
    epochs_drop = 10
    lr = initial_lr * (drop ** (np.floor(1 + epoch) / epochs_drop))
    return max(lr, 0.0)
```

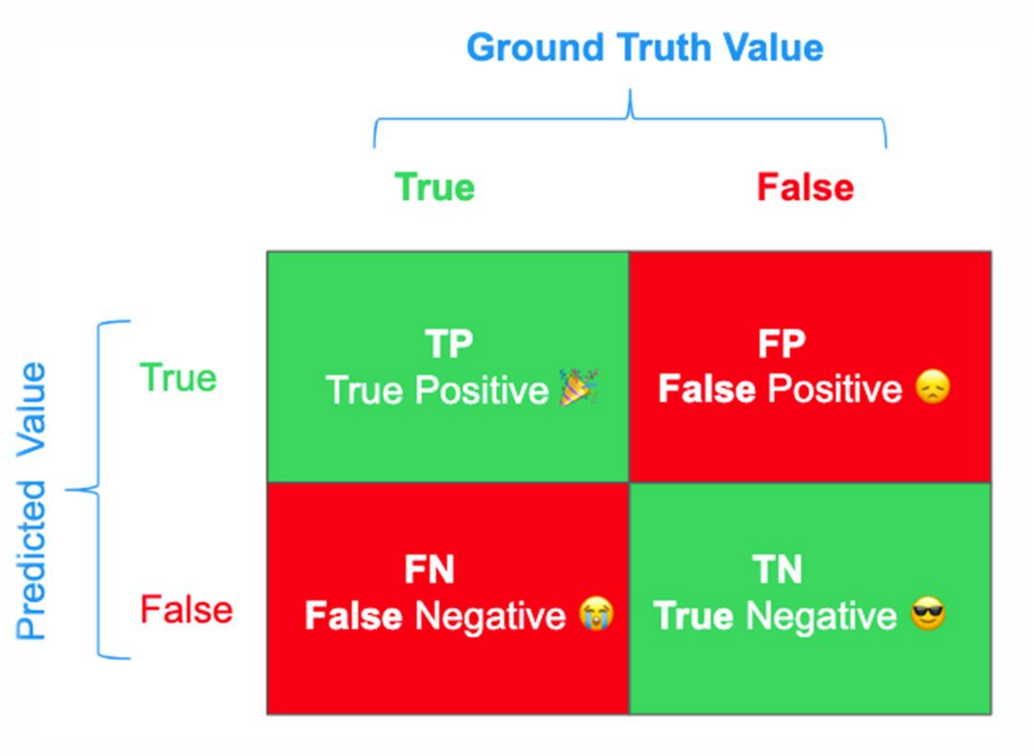
About Metrics

$$\boldsymbol{Precision} = \frac{TP}{TP + FP}$$

$$\boldsymbol{Recall} = \frac{TP}{TP + FN}$$

$$\boldsymbol{Accuracy} = \frac{TP + TN}{TP + FP + TN + FN}$$

$$\boldsymbol{F1 - Score} = \frac{2\ \times Precision\ \times Recall}{Precision + Recall}$$

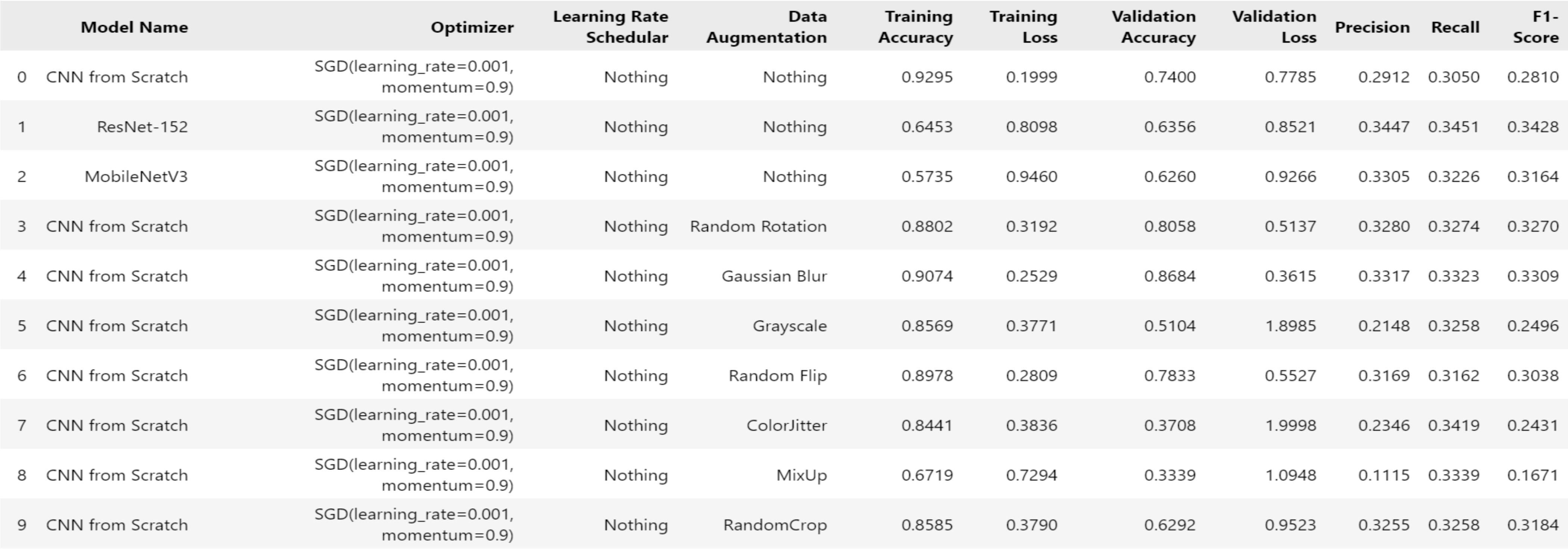


Ground Truth Value

Predicted Value	True	False
True	**TP** True Positive 🎉	**FP** False Positive 😞
False	**FN** False Negative 😭	**TN** True Negative 😎

Show & Save Results

```python
def plot_metrics_and_matrix(history, model, validation_generator, title, task):
```

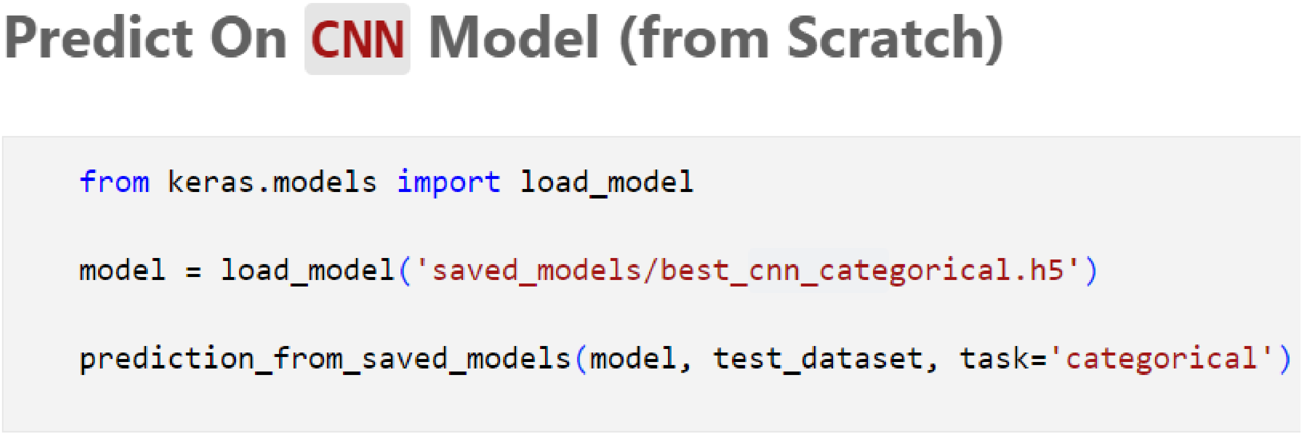


	Model Name	Optimizer	Learning Rate Schedular	Data Augmentation	Training Accuracy	Training Loss	Validation Accuracy	Validation Loss	Precision	Recall	F1-Score
0	CNN from Scratch	SGD(learning_rate=0.001, momentum=0.9)	Nothing	Nothing	0.9295	0.1999	0.7400	0.7785	0.2912	0.3050	0.2810
1	ResNet-152	SGD(learning_rate=0.001, momentum=0.9)	Nothing	Nothing	0.6453	0.8098	0.6356	0.8521	0.3447	0.3451	0.3428
2	MobileNetV3	SGD(learning_rate=0.001, momentum=0.9)	Nothing	Nothing	0.5735	0.9460	0.6260	0.9266	0.3305	0.3226	0.3164
3	CNN from Scratch	SGD(learning_rate=0.001, momentum=0.9)	Nothing	Random Rotation	0.8802	0.3192	0.8058	0.5137	0.3280	0.3274	0.3270
4	CNN from Scratch	SGD(learning_rate=0.001, momentum=0.9)	Nothing	Gaussian Blur	0.9074	0.2529	0.8684	0.3615	0.3317	0.3323	0.3309
5	CNN from Scratch	SGD(learning_rate=0.001, momentum=0.9)	Nothing	Grayscale	0.8569	0.3771	0.5104	1.8985	0.2148	0.3258	0.2496
6	CNN from Scratch	SGD(learning_rate=0.001, momentum=0.9)	Nothing	Random Flip	0.8978	0.2809	0.7833	0.5527	0.3169	0.3162	0.3038
7	CNN from Scratch	SGD(learning_rate=0.001, momentum=0.9)	Nothing	ColorJitter	0.8441	0.3836	0.3708	1.9998	0.2346	0.3419	0.2431
8	CNN from Scratch	SGD(learning_rate=0.001, momentum=0.9)	Nothing	MixUp	0.6719	0.7294	0.3339	1.0948	0.1115	0.3339	0.1671
9	CNN from Scratch	SGD(learning_rate=0.001, momentum=0.9)	Nothing	RandomCrop	0.8585	0.3790	0.6292	0.9523	0.3255	0.3258	0.3184

Trained Model

- I Trained 35 models at all
- The Best model in task 1 is with **Adam** Optimizer and **LinearLR** **-** CNN from scratch
- The Best model in task 2 is with **SGD** and **LinearLR** - CNN from scratch

- **At the end I saved 6 models with weights to predict on the test data.**



Predict On CNN Model (from Scratch)

```
from keras.models import load_model

model = load_model('saved_models/best_cnn_categorical.h5')

prediction_from_saved_models(model, test_dataset, task='categorical')
```

Refrences

https://chat.openai.com/

https://keras.io/

https://www.tensorflow.org/

https://www.geeksforgeeks.org/

ADS-Course (Dr. Salavati)

https://towardsdatascience.com/

Thank you for your attention