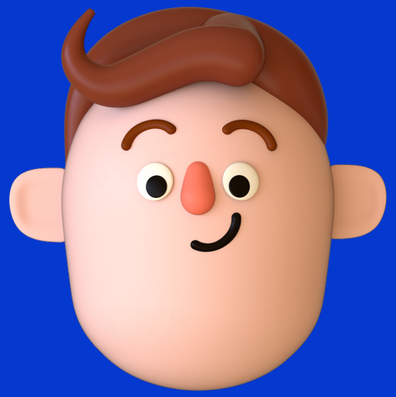
UKRIDA
Universitas Kristen Krida Wacana

MACHINE LEARNING

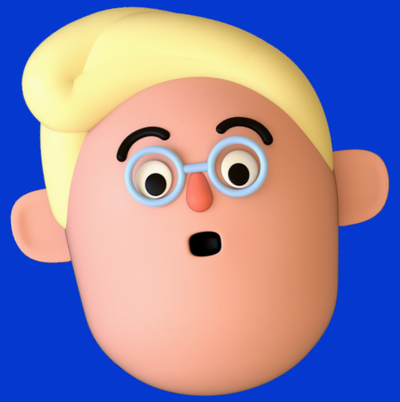
HW#1 - Indonsesian Street Food Classification

KELOMPOK 4



ALBERT SAPUTRA ZEBUA

412020012

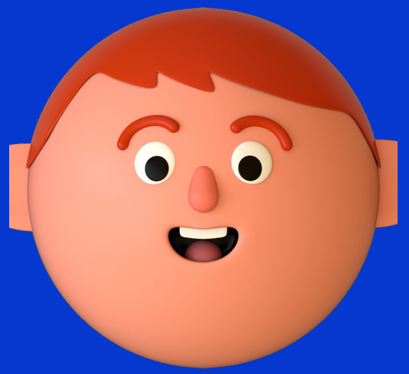


MAHESA RAFIAN SYAH

412020016

JONATAN SIANTURI

412020027

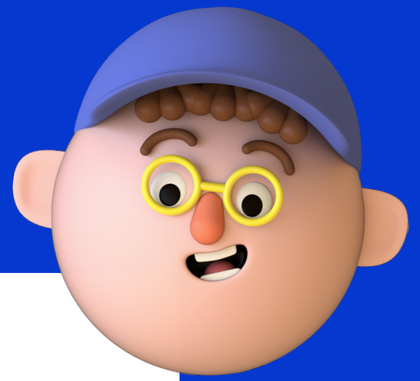


CALVIN JEDI NUGRAHA

412020030

UKRIDA
Universitas Kristen Krida Wacana

DATASET

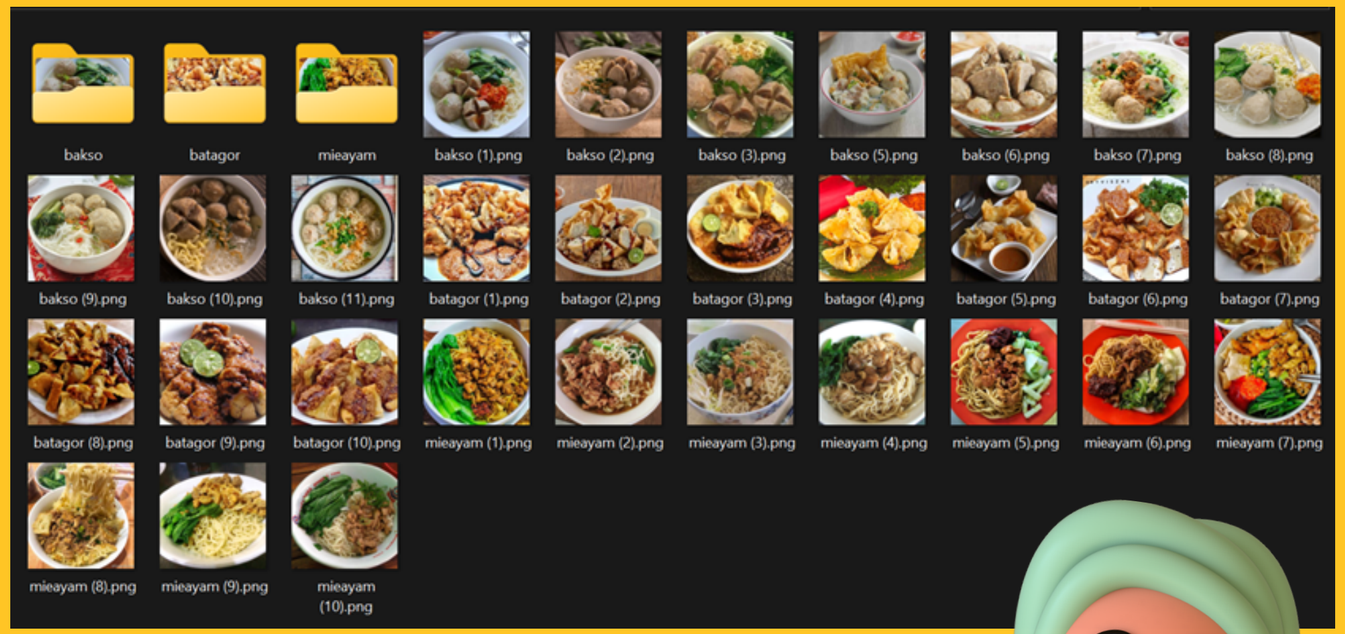


UKRIDA
Universitas Kristen Krida Wacana

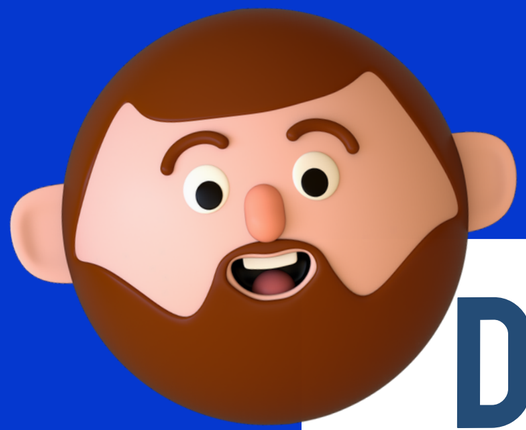
DATASET

Berikut adalah dataset Indonesian Street Food, terdiri atas Batagor, Bakso, dan Mie Ayam.

Sumber dataset sebagian besar diambil menggunakan foto asli dari makanan-makanan yang dipilih



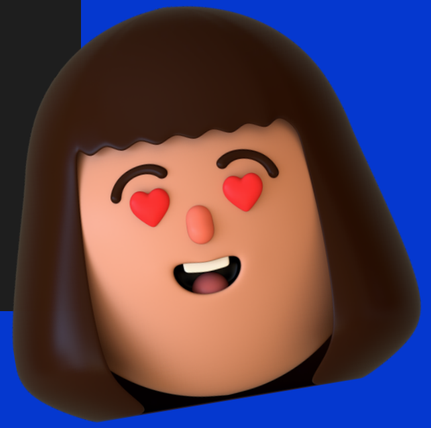
UKRIDA
Universitas Kristen Krida Wacana



DATALOADER

INF-UKKW-2023-ML-DL > MyUtils > MyDatasets.py > ...

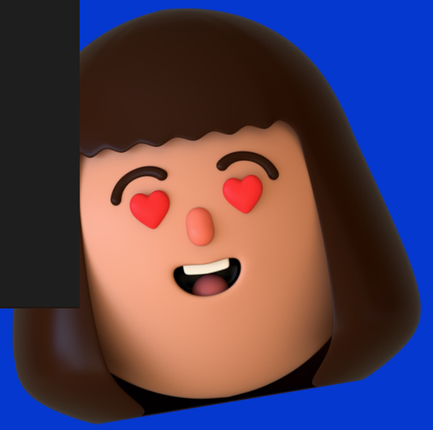
```python
import numpy as np
import os
from PIL import Image
import torch
from torch.utils.data import Dataset, DataLoader
import torchvision.transforms as transforms
import albumentations as A
from albumentations.pytorch import ToTensorV2

both_transform = A.Compose(
    [A.Resize(width=512, height=512),], additional_targets={"image0": "image"},
)

transform_only_input = A.Compose(
    [
        A.HorizontalFlip(p=0.5),
        A.ColorJitter(p=0.2),
        A.Normalize(mean=[0.5, 0.5, 0.5], std=[0.5, 0.5, 0.5], max_pixel_value=255.0,),
        ToTensorV2(),
    ]
)

transform_only_mask = A.Compose(
    [
        A.Normalize(mean=[0.5, 0.5, 0.5], std=[0.5, 0.5, 0.5], max_pixel_value=255.0,),
        ToTensorV2(),
    ]
)
```



UKRIDA
Universitas Kristen Krida Wacana

INF-UKKW-2023-ML-DL > MyUtils > MyDatasets.py > ...

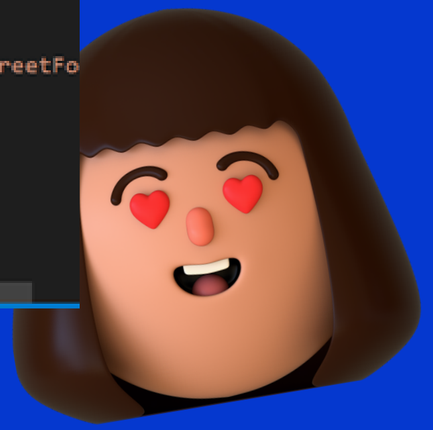
```python
class UkridaDataset(Dataset):
    def __init__(self, root_dir):
        self.root_dir = root_dir
        self.list_files = os.listdir(self.root_dir)
        self.list_files = [i for i in self.list_files if i.endswith(".png")]

    def __len__(self):
        return len(self.list_files)

    def __getitem__(self, index):
        img_file = self.list_files[index]
        img_path = os.path.join(self.root_dir, img_file)
        image = np.array(Image.open(img_path))
        image = image[:, :, :3]  # Convert RGBA to RGB

        w_img = image.shape[1]
        w_img_half = w_img // 2
        input_image = image[:, :w_img_half, :]
        target_image = image[:, w_img_half:, :]

        augmentations = both_transform(image=input_image, image0=target_image)
        input_image = augmentations["image"]
        target_image = augmentations["image0"]

        input_image = transform_only_input(image=input_image)["image"]
        target_image = transform_only_mask(image=target_image)["image"]

        return input_image, target_image
```



UKRIDA
Universitas Kristen Krida Wacana

INF-UKKW-2023-ML-DL > MyUtils > MyDatasets.py > ...

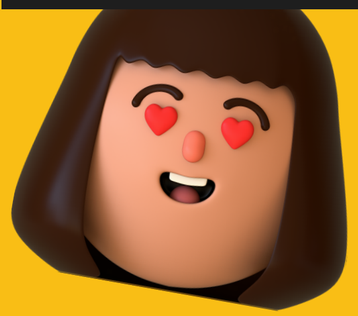
```
56
57          return input_image, target_image
58
59
60  class CustomDataLoader:
61      def __init__(self, root_dir, batch_size=1, shuffle=True):
62          self.dataset = UkridaDataset(root_dir)
63          self.dataloader = DataLoader(
64              self.dataset,
65              batch_size=batch_size,
66              shuffle=shuffle
67          )
68
69      def __iter__(self):
70          return iter(self.dataloader)
71
72      def __len__(self):
73          return len(self.dataloader)
74
75
76  # test case
77  if __name__ == "__main__":
78      dataloader = CustomDataLoader("C:/Users/Asus/ML-2023/INF-UKKW-2023-ML-DL/data/IndonesianStreetFo
79      for x, y in dataloader:
80          print(x.shape)
81          import sys
82          sys.exit()
83
```



UKRIDA
Universitas Kristen Krida Wacana

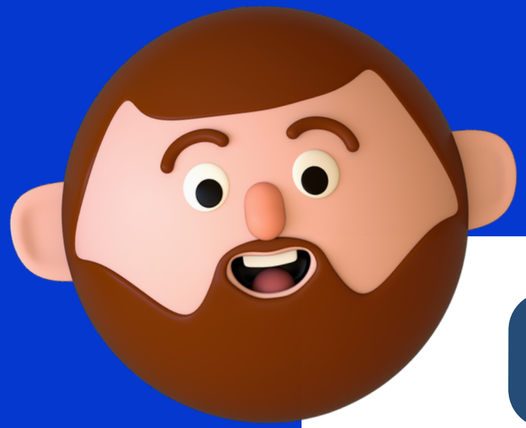
OUTPUT

```
PROBLEMS 62   OUTPUT   TERMINAL   COMMENTS   DEBUG CONSOLE

(base) PS C:\Users\Asus\ML-2023\INF-UKKW-2023-ML-DL\MyUtils> python MyDatasets.py
torch.Size([1, 3, 512, 512])
(base) PS C:\Users\Asus\ML-2023\INF-UKKW-2023-ML-DL\MyUtils>
```



- In PyTorch, tensors are multi-dimensional arrays that can hold numerical data.
- So, the given tensor shape indicates a single batch of RGB images with a resolution of 512x512 pixels.

UKRIDA
Universitas Kristen Krida Wacana



CNN MODEL

UKRIDA
Universitas Kristen Krida Wacana

```
        return x


def train(model, train_loader, criterion, optimizer):
    model.train()
    running_loss = 0.0
    for i, data in enumerate(train_loader, 0):
        inputs, labels = data
        optimizer.zero_grad()
        outputs = model(inputs)
        loss = criterion(outputs, labels)
        loss.backward()
        optimizer.step()
        running_loss += loss.item()
    train_loss = running_loss / len(train_loader)
    return train_loss


def test(model, test_loader):
    model.eval()
    correct = 0
    total = 0
    with torch.no_grad():
        for data in test_loader:
            images, labels = data
            outputs = model(images)
            _, predicted = torch.max(outputs.data, 1)
            total += labels.size(0)
            correct += (predicted == labels).sum().item()
```

INF-UKKW-2023-ML-DL > core_model > part_1_model.py > main

```
        return x


def train(model, train_loader, criterion, optimizer):
    model.train()
    running_loss = 0.0
    for i, data in enumerate(train_loader, 0):
        inputs, labels = data
        optimizer.zero_grad()
        outputs = model(inputs)
        loss = criterion(outputs, labels)
        loss.backward()
        optimizer.step()
        running_loss += loss.item()
    train_loss = running_loss / len(train_loader)
    return train_loss


def test(model, test_loader):
    model.eval()
    correct = 0
    total = 0
    with torch.no_grad():
        for data in test_loader:
            images, labels = data
            outputs = model(images)
            _, predicted = torch.max(outputs.data, 1)
            total += labels.size(0)
            correct += (predicted == labels).sum().item()
```

UKRIDA
Universitas Kristen Krida Wacana

INF-UKKW-2023-ML-DL > core_model > part_1_model.py > main

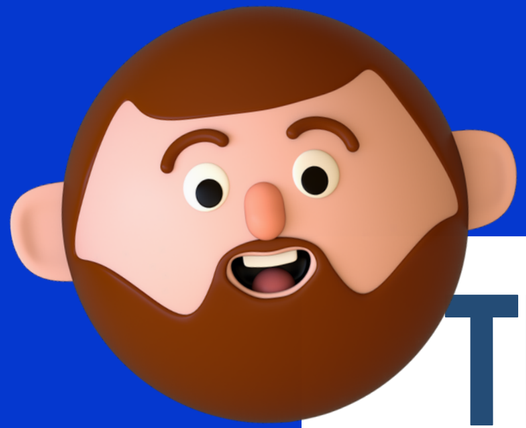
```
                total += labels.size(0)
                correct += (predicted == labels).sum().item()
        test_accuracy = correct / total
        return test_accuracy


def main():
    # Set random seed for reproducibility
    torch.manual_seed(2023)

    # Define transformations for the dataset
    transform = transforms.Compose([
        transforms.Resize((128, 128)),
        transforms.ToTensor(),
        transforms.Normalize((0.5, 0.5, 0.5), (0.5, 0.5, 0.5))
    ])

    # Load the training dataset
    train_dataset_path = "C:/Users/Asus/ML-2023/INF-UKKW-2023-ML-DL/data/IndonesianStreetFood/train"
    train_dataset = torchvision.datasets.ImageFolder(train_dataset_path, transform=transform)
    train_loader = torch.utils.data.DataLoader(train_dataset, batch_size=30, shuffle=True, num_workers=2)

    # Load the test dataset
    test_dataset_path = "C:/Users/Asus/ML-2023/INF-UKKW-2023-ML-DL/data/IndonesianStreetFood/test"
    test_dataset = torchvision.datasets.ImageFolder(test_dataset_path, transform=transform)
    test_loader = torch.utils.data.DataLoader(test_dataset, batch_size=30, shuffle=False, num_workers=2)

    # Create the model
    model = YourOwnCNN()
```

UKRIDA
Universitas Kristen Krida Wacana

INF-UKKW-2023-ML-DL > core_model > part_1_model.py > main

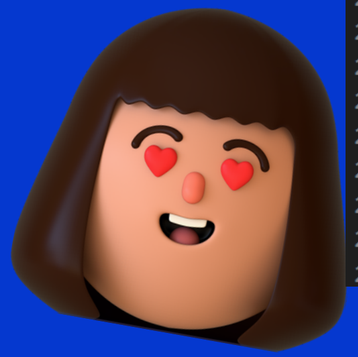
```python
    train_loader = torch.utils.data.DataLoader(train_dataset, batch_size=30, shuffle=True, num_workers=2)

    # Load the test dataset
    test_dataset_path = "C:/Users/Asus/ML-2023/INF-UKKW-2023-ML-DL/data/IndonesianStreetFood/test"
    test_dataset = torchvision.datasets.ImageFolder(test_dataset_path, transform=transform)
    test_loader = torch.utils.data.DataLoader(test_dataset, batch_size=30, shuffle=False, num_workers=2)

    # Create the model
    model = YourOwnCNN()

    # Define the loss function and optimizer
    criterion = nn.CrossEntropyLoss()
    optimizer = optim.SGD(model.parameters(), lr=0.001, momentum=0.9)

    # Train the model
    for epoch in range(10):
        train_loss = train(model, train_loader, criterion, optimizer)
        print(f"Epoch {epoch+1}/{10}, Training Loss: {train_loss}")

    # Test the model
    test_accuracy = test(model, test_loader)
    print(f"Test Accuracy: {test_accuracy}")


if __name__ == '__main__':
    main()
```

UKRIDA
Universitas Kristen Krida Wacana



TRAIN & TEST

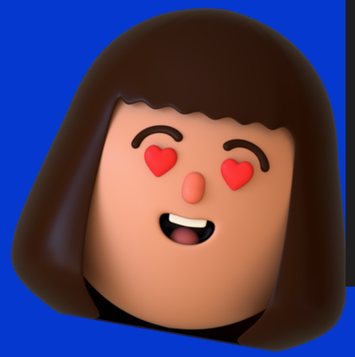
UKRIDA
Universitas Kristen Krida Wacana



NF-UKKW-2023-ML-DL > train_CNN.py > YourOwnCNN > __init__

```python
import torch
import torch.nn as nn
import torch.optim as optim
import torchvision
import torchvision.transforms as transforms


class YourOwnCNN(nn.Module):
    def __init__(self):
        super(YourOwnCNN, self).__init__()
        self.conv1 = nn.Conv2d(3, 6, 5)
        self.pool1 = nn.MaxPool2d(2, 2)
        self.conv2 = nn.Conv2d(6, 16, 5)
        self.pool2 = nn.MaxPool2d(2, 2)
        self.fc1 = nn.Linear(16 * 29 * 29, 120)
        self.fc2 = nn.Linear(120, 84)
        self.fc3 = nn.Linear(84, 3)

    def forward(self, x):
        x = nn.ReLU()(self.conv1(x))
        x = self.pool1(x)
        x = nn.ReLU()(self.conv2(x))
        x = self.pool2(x)
        x = x.view(x.size(0), -1)
        x = nn.ReLU()(self.fc1(x))
        x = nn.ReLU()(self.fc2(x))
        x = self.fc3(x)
        return x
```

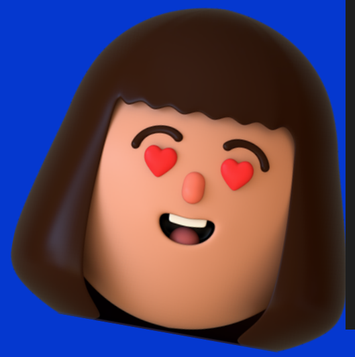
UKRIDA
Universitas Kristen Krida Wacana



NF-UKKW-2023-ML-DL > train_CNN.py > YourOwnCNN > __init__

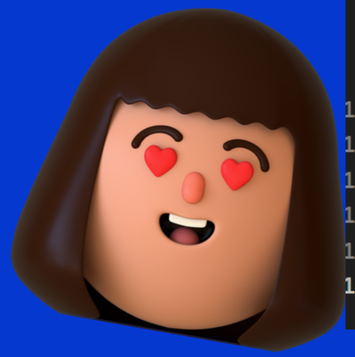
```
29
30
31  def train(model, train_loader, criterion, optimizer):
32      model.train()
33      running_loss = 0.0
34      for i, data in enumerate(train_loader, 0):
35          inputs, labels = data
36          optimizer.zero_grad()
37          outputs = model(inputs)
38          loss = criterion(outputs, labels)
39          loss.backward()
40          optimizer.step()
41          running_loss += loss.item()
42          if i % 100 == 99:  # Print loss every 100 mini-batches
43              print(f"Epoch {epoch+1}/{10}, Mini-batch {i+1}, Loss: {running_loss / 100}")
44              running_loss = 0.0
45      train_loss = running_loss / len(train_loader)
46      return train_loss
47
48
49  def test(model, test_loader):
50      model.eval()
51      correct = 0
52      total = 0
53      with torch.no_grad():
54          for data in test_loader:
55              images, labels = data
56              outputs = model(images)
```

UKRIDA
Universitas Kristen Krida Wacana



```
train_CNN.py 6, U | README.md | part_2_model.py 9+ | train_dcgan.py 9 | requirements.txt
NF-UKKW-2023-ML-DL > train_CNN.py > ...
56                  outputs = model(images)
57                  _, predicted = torch.max(outputs.data, 1)
58                  total += labels.size(0)
59                  correct += (predicted == labels).sum().item()
60      test_accuracy = correct / total
61      return test_accuracy
62
63
64  def main():
65      # Set random seed for reproducibility
66      torch.manual_seed(2023)
67
68      # Define transformations for the dataset
69      transform = transforms.Compose([
70          transforms.Resize((128, 128)),
71          transforms.ToTensor(),
72          transforms.Normalize((0.5, 0.5, 0.5), (0.5, 0.5, 0.5))
73      ])
74
75      # Load the training dataset
76      train_dataset_path = "C:/Users/Asus/ML-2023/INF-UKKW-2023-ML-DL/data/IndonesianStreetFood/train"
77      train_dataset = torchvision.datasets.ImageFolder(train_dataset_path, transform=transform)
78      train_loader = torch.utils.data.DataLoader(train_dataset, batch_size=30, shuffle=True, num_workers=2)
79
80      # Load the test dataset
81      test_dataset_path = "C:/Users/Asus/ML-2023/INF-UKKW-2023-ML-DL/data/IndonesianStreetFood/test"
82      test_dataset = torchvision.datasets.ImageFolder(test_dataset_path, transform=transform)
83      test_loader = torch.utils.data.DataLoader(test_dataset, batch_size=30, shuffle=False, num_workers=2)
84
```

UKRIDA
Universitas Kristen Krida Wacana



el.py 6, M | train_CNN.py 6, U | README.md | part_2_model.py 9+ | train_dcgan.py 9 | requirements.txt

NF-UKKW-2023-ML-DL > train_CNN.py > ...

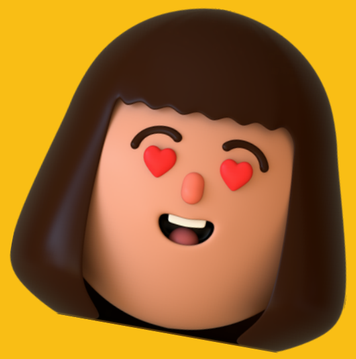
```python
    # Load the test dataset
    test_dataset_path = "C:/Users/Asus/ML-2023/INF-UKKW-2023-ML-DL/data/IndonesianStreetFood/test"
    test_dataset = torchvision.datasets.ImageFolder(test_dataset_path, transform=transform)
    test_loader = torch.utils.data.DataLoader(test_dataset, batch_size=30, shuffle=False, num_workers=2)

    # Create the model
    model = YourOwnCNN()

    # Define the loss function and optimizer
    criterion = nn.CrossEntropyLoss()
    optimizer = optim.SGD(model.parameters(), lr=0.001, momentum=0.9)

    # Train the model
    for epoch in range(10):
        train_loss = train(model, train_loader, criterion, optimizer)
        print(f"Epoch {epoch+1}/{10}, Training Loss: {train_loss}")

    # Test the model
    test_accuracy = test(model, test_loader)
    formatted_accuracy = "{:.2%}".format(test_accuracy)  # Format test accuracy with two digits in front of th
    print(f"Test Accuracy: {formatted_accuracy}")


if __name__ == '__main__':
    main()

```

UKRIDA
Universitas Kristen Krida Wacana

OUTPUT

```
(base) PS C:\Users\Asus\ML-2023\INF-UKKW-2023-ML-DL> python train_CNN.py
Epoch 1/10, Training Loss: 1.100756287574768
Epoch 2/10, Training Loss: 1.1006642580032349
Epoch 3/10, Training Loss: 1.1004910469055176
Epoch 4/10, Training Loss: 1.1002503633499146
Epoch 5/10, Training Loss: 1.099954605102539
Epoch 6/10, Training Loss: 1.0996112823486328
Epoch 7/10, Training Loss: 1.0992361307144165
Epoch 8/10, Training Loss: 1.0988430976867676
Epoch 9/10, Training Loss: 1.0984376668930054
Epoch 10/10, Training Loss: 1.0980299711227417
Model weights saved at: C:/Users/Asus/ML-2023/INF-UKKW-2023-ML-DL/core_model/model_weights.pth
Training completed.
```

```
(base) PS C:\Users\Asus\ML-2023\INF-UKKW-2023-ML-DL> python test_CNN.py
Test Accuracy: 36.67%
```

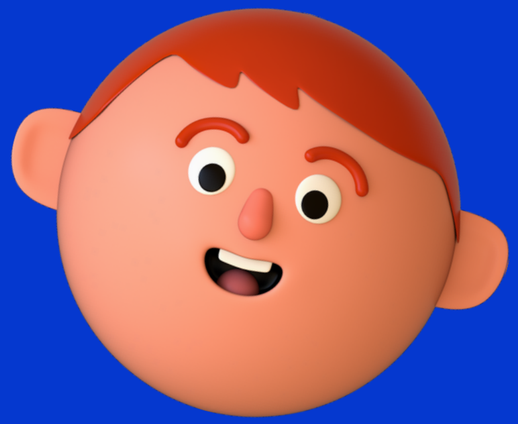


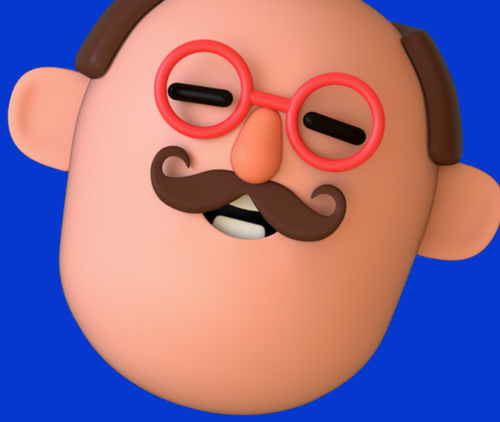
Total Epochs:
10 Epoch

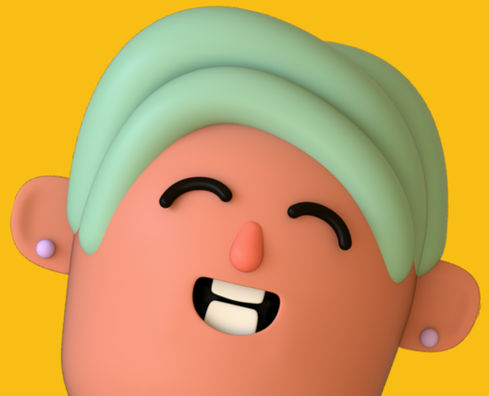
Test Accuracy:
36.67%

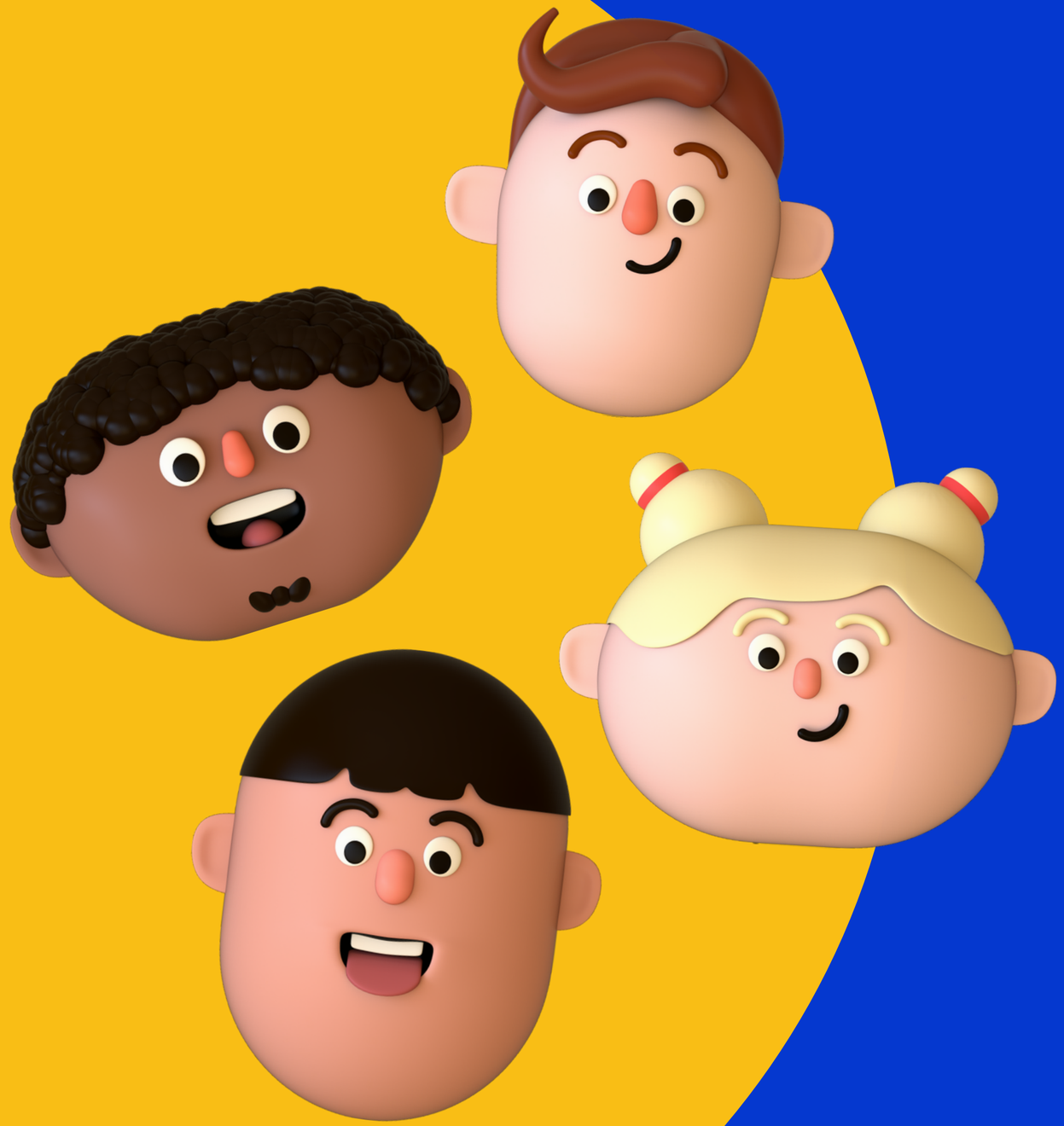
UKRIDA
Universitas Kristen Krida Wacana

ANY QUESTION?









THANK YOU!

UKRIDA
Universitas Kristen Krida Wacana