Yeditepe University
Faculty of Engineering
Electrical and Electronics Engineering
EE492 Engineering Project Progress Report

Name Surname of the Student	Muhammet Hakan Taştan
Student ID	20190701009
Engineering Project Title	Facial Emotion Recognition System

Brief Summary of the Engineering Project Subject and Aims of the Project:

This project aims to develop a Facial Emotion Recognition System using the AffectNet database. Model architectures will be developed to compare its performance against both a widely-used model and a state-of-the-art model in the field. The ultimate goal is to enhance human-computer interaction through better emotion-recognition systems.

The tasks that have been completed until now:

1. Literature review
2. Acquired access to the AffectNet dataset.
3. Conducted exploratory data analysis to understand the dataset structure and identify the problems.
4. Defined the architectures of the facial emotion recognition models.

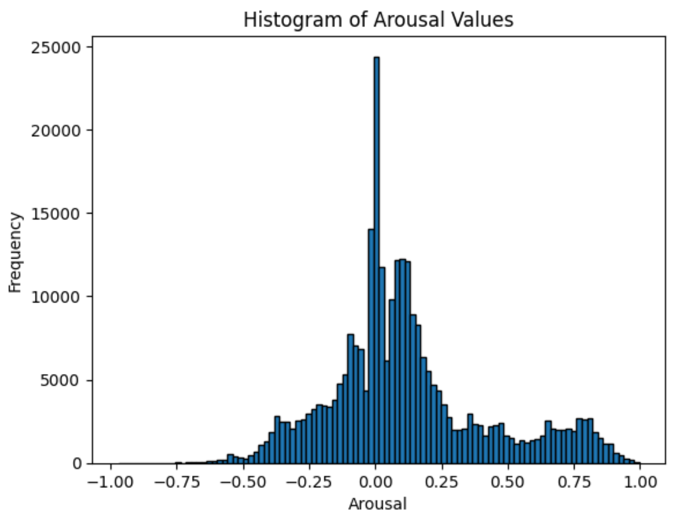
Problems - unusual situations that has been experienced until now:

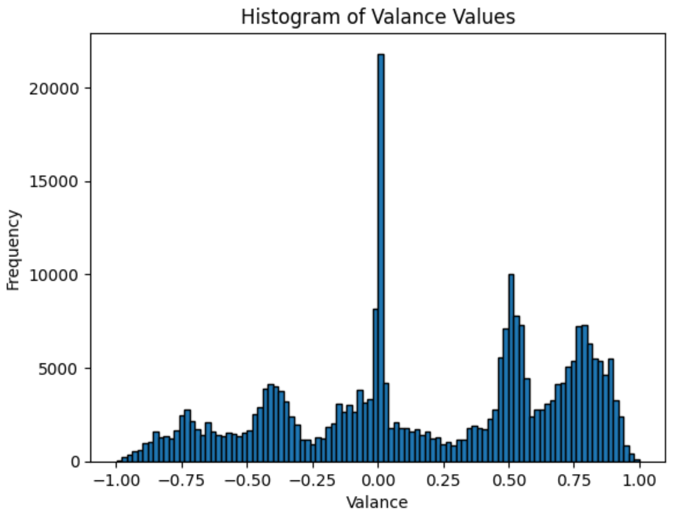
1. Dataset Imbalance:
   The first problem is the unbalanced data in the dataset, with certain emotion labels having less values compared to others. This could impact the model's overall accuracy negatively. To address this issue, oversampling techniques for the labels with lower representation will be used in the dataset to ensure a balanced training set.
2. Computational Power Limitations:
   Training deep neural networks needs considerable computational resources, which is a limiting factor. To overcome it, cloud computing platforms such as Google Collab, Kaggle, etc., are utilized.
3. Real-time Inference Speed:
   The trained models not only predict accurately but also perform it in a short amount of time. To increase the performance and speed of the model, some considerations need to be made. One planned solution is utilizing convolution factorizations.

Plans for the rest of the project:

1. Oversampling Implementation
2. Continue to train models
3. Fine-tuning and Optimization
4. Documentation and Reporting

Attachments (If any):






```
class AffectNet(Dataset):
    def __init__(self, annotations_file, root_dir, transform=None,
target_transform=None):
        self.annotations = pd.read_csv(annotations_file)
        self.root_dir = root_dir
        self.transform = transform
        self.target_transform = target_transform

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

    def __getitem__(self, idx):
        img_path = os.path.join(self.root_dir,
self.annotations.iloc[idx, 0])
        image = read_image(img_path)

        labels = self.annotations.iloc[idx, 1:]
        labels = labels.to_numpy()
        labels = torch.from_numpy(labels.astype('float'))
        labels = labels.to(torch.float32)

        if self.transform:
            image = self.transform(image)
        if self.target_transform:
            labels = self.target_transform(labels)

        return image, labels
```

```
# Define device, loss function and optimizer
device = torch.device("cuda") if torch.cuda.is_available() else
torch.device("cpu")
loss_fn = nn.L1Loss()
optimizer = torch.optim.Adam(model.parameters(), lr=0.001)

# Move model to the device
model.to(device)

# Number of epochs
epochs = 3

# Training loop
for epoch in range(epochs):
    running_loss = 0.0
    for step, [inputs, targets] in enumerate(train_dataloader):
        # get the inputs; data is a list of [inputs, labels]
        inputs = inputs.to(device)
        targets = targets.to(device)

        # zero the parameter gradients
        optimizer.zero_grad()

        # forward + backward + optimize
        preds = model(inputs)
        loss = loss_fn(preds, targets)
        loss.backward()
        optimizer.step()

        # print statistics
        running_loss += loss.item()
        # Log every 100 batches.
        if step % 100 == 0:
            print(
                f"Training loss (for 1 batch) at step {step}:
{loss.detach().to('cpu').numpy():.4f}"
            )
            print(f"Seen so far: {(step + 1) * 32} samples")

print('Finished Training')
```