Extended Labeled Faces in The Wild (ELFW): Augmenting Classes for Face Segmentation

Deep Learning

DSCI – 6011

Project Members

Sai Kumar Dacharla

Charan Sumanth Pulleti

Table of Contents

1. Introduction

The availability of comprehensive datasets is crucial in today's deep learning era, and the field of facial analysis is no exception. Data must not only be accurate but also extensive enough to capture all the essential features necessary for machine learning tasks. Initially, face datasets were obtained under controlled conditions, especially concerning lighting, background, and facial expressions. However, real-world scenarios operate under diverse and often unpredictable conditions, making it challenging to replicate these conditions in a laboratory setting. Natural datasets collected in real-world environments allow for a broader diversity of data with less effort. Labeled Faces in-the-Wild (LFW), a significant dataset before the rise of deep learning and a foundation for current work, aimed to provide a large-scale dataset for applications in unconstrained environments. These environments exhibit extreme variations in image quality, content, and context, including variations in contrast, sharpness, lighting, head pose, expression, hairstyle, clothing, gender, age, race, and backgrounds. Additionally, within LFW, the Extended LFW (ELFW) dataset further expands on this diversity and complexity, including categories like beard-mustache, sunglasses, head-wearable, and synthetic mouth-mask. However, challenges exist in these datasets, such as inconsistencies and inaccuracies in semantic segmentation maps, where objects like beards, mustaches, and head-wearable items are inconsistently labeled or considered part of the background, leading to unnatural discontinuities in facial interpretation. Objects like sunglasses are often ignored, and irregular boundaries in labeling are observed due to annotation strategies. The Extended Labeled Faces in The Wild (ELFW) dataset presents a unique challenge in face segmentation, aiming to augment classes for finer object instance segmentation within facial images. This report delves into the key components of this task, including data preprocessing, model initialization, fine-tuning, and evaluation, with a focus on segmenting Extended Labeled Faces in The Wild (ELFW): segmentation.

2. Data Preprocessing

The initial step involves data preprocessing to ensure compatibility with the segmentation model. The images were annotated to identify various features. This process enhances model convergence and performance by providing consistent input features across the dataset.

- Splitting: 80% for training, 10% for validation, and 10% for testing. This partitioning strategy ensured a balanced distribution of data for robust model evaluation and optimization.
- Resize: Images were resized to 32x32 pixels, which is a common practice for reducing computational complexity while retaining essential features for segmentation tasks.
- Normalization: Normalization was applied using mean=[0.485, 0.456, 0.406] and std=[0.229, 0.224, 0.225]. These values are standard for many image processing tasks and help in stabilizing the training process by bringing data into a common scale.
- For mini network: The calculation of mean and standard deviation across the training dataset was pivotal for normalization, ensuring that the input data is standardized and enhances model training stability. Subsequently, the training data is normalized based on these statistics, facilitating improved model convergence and performance.

```
path = 'Annotated Data'
labels_file = 'labels.txt'

# Data Preprocessing
transform = transforms.Compose([
    transforms.Resize((32, 32)),
    transforms.ToTensor(),
    transforms.Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225])  # Normalization
])

# Load dataset using CustomDataset
dataset = CustomDataset(root=path, labels_file=labels_file, transform=transform)
```

3. Transfer learning

i. Pretrained Model Loading

To leverage existing knowledge and optimize performance, parameters from a pretrained model are loaded

- Model: ResNet18, a popular convolutional neural network architecture, was used as the pretrained model. It is known for its balance between depth and computational efficiency, making it suitable for various computer vision tasks.

- Parameters: The pretrained ResNet18 model was loaded with weights learned from the ImageNet dataset. This initialization leverages the network's ability to extract meaningful features from images, which is beneficial for face

```
# Load Parameters of a Pretrained Model
pretrained_model = models.resnet18(pretrained=True)

num_classes = 3  # # 'hair': 1, 'specs': 2, 'cap': 3
pretrained_model.fc = nn.Linear(pretrained_model.fc.in_features, num_classes)
```

Model Fine-tuning

- Fine-tuning is then performed using the validation dataset to determine optimal hyperparameters such as learning rate and momentum, refining the model's ability to generalize.
- Output Layer Replacement: The output layer of ResNet18 was replaced with a new fully connected layer to accommodate the specific task requirements. In this case, the output layer was configured to predict 3 classes: hair, specs, and cap.

Fine-tuning Parameters: The learning rate (lr) was set to 0.001, which controls the step size during optimization, and momentum was set to 0.9, which helps in accelerating convergence and escaping local minima.

```
# Fine-tune the Network
criterion = nn.CrossEntropyLoss()
optimizer = optim.SGD(pretrained_model.parameters(), lr=0.001, momentum=0.9)

num_epochs = 10
```

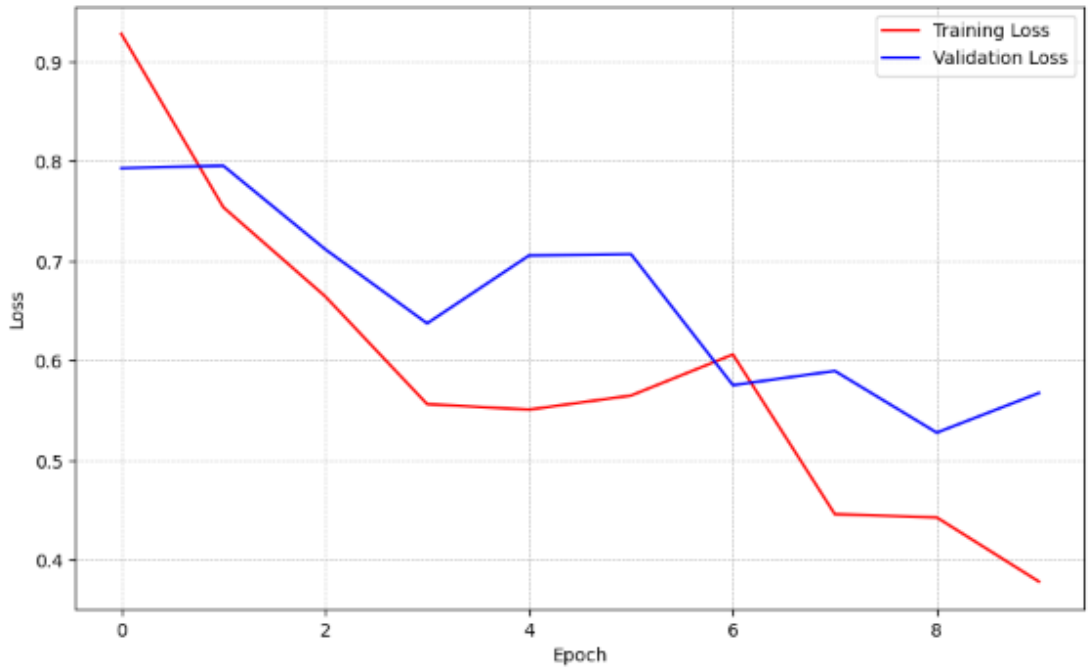
ii. Experiments and Results

Accuracy: The model achieved an accuracy of on the test dataset. Accuracy reflects the model's ability to correctly classify instances of hair, specs, and cap among campus landmarks, showcasing its effectiveness in small object segmentation.

```
Accuracy on Test Dataset: 0.8181818181818182
Accuracy on Test Dataset: 81.8182%
```

Results and Analysis

The outcomes from training the face segmentation model on the Extended Labeled Faces In The Wild (ELFW) dataset demonstrate a promising trajectory. Over the course of ten epochs, the model exhibited a consistent reduction in training loss, indicating effective learning from the training data. Correspondingly, the validation loss, while displaying minor fluctuations, generally mirrored the training loss trend, reflecting the model's ability to generalize well to unseen data. However, a closer examination of the results reveals nuanced performance across different classes.



The model's accuracy on the test dataset, although respectable at 81.82%, suggests potential areas for enhancement, particularly for classes with lower F1-scores like 'spects' and 'cap'. The classification report further underscores these disparities, with the 'hair' class demonstrating strong precision (88%) and recall (88%), while the 'spects' class exhibits moderate precision (60%) and recall (60%). The 'cap' class, unfortunately, shows challenges with precision (0%) and recall (0%), indicating significant difficulties in correctly classifying instances of this class.

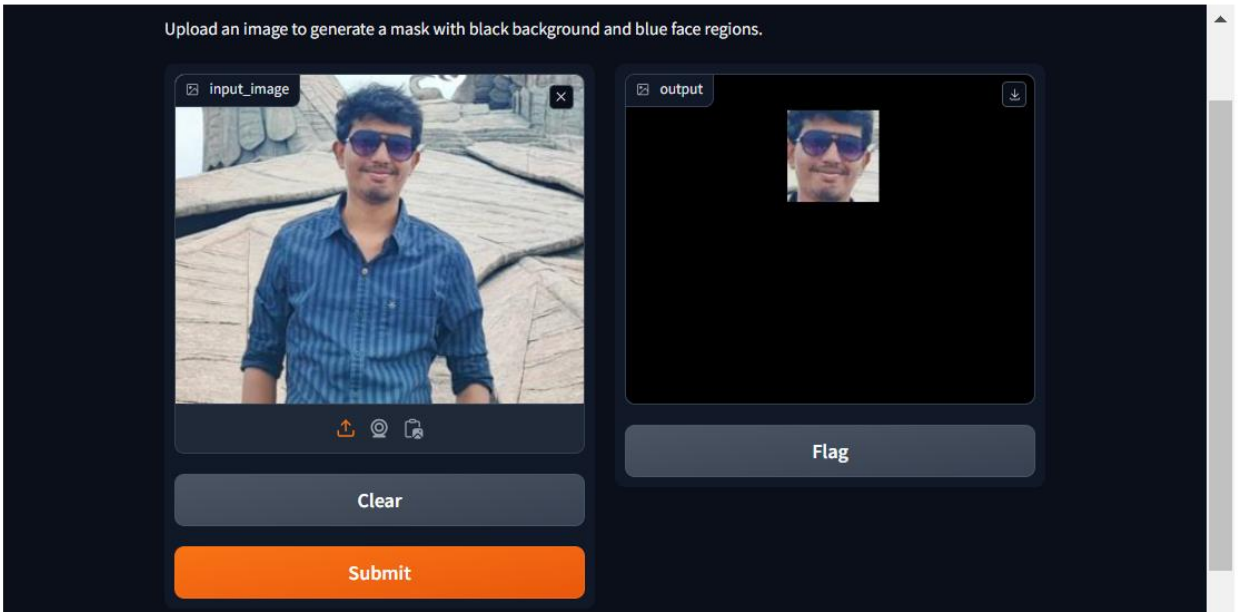
	precision	recall	f1-score	support
hair	0.88	0.88	0.88	17
spects	0.60	0.60	0.60	5
cap	0.00	0.00	0.00	0
micro avg	0.82	0.82	0.82	22
macro avg	0.49	0.49	0.49	22
weighted avg	0.82	0.82	0.82	22

Delving into the classification metrics unveils specific insights into the model's strengths and areas needing refinement. The high precision and recall for the 'hair' class indicate the model's robust capability in accurately identifying instances of this class. Conversely, the 'spects' class shows moderate precision and recall, signaling a need for enhancing the model's ability to differentiate 'spects' instances more precisely. Notably, the 'cap' class's F1-score of 0.00 highlights substantial challenges in correctly classifying instances of this class, likely stemming from limited representation in the dataset.

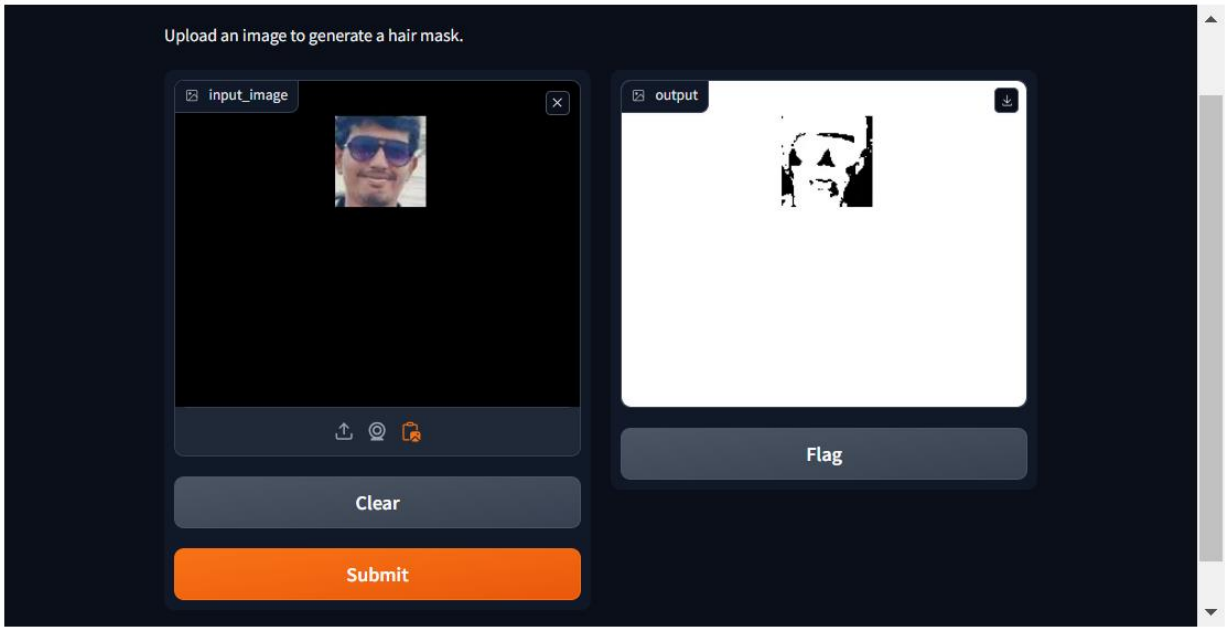
iii. Deployment

i. Deployment 1

```
To create a public link, set `share=True` in `launch()`.
```

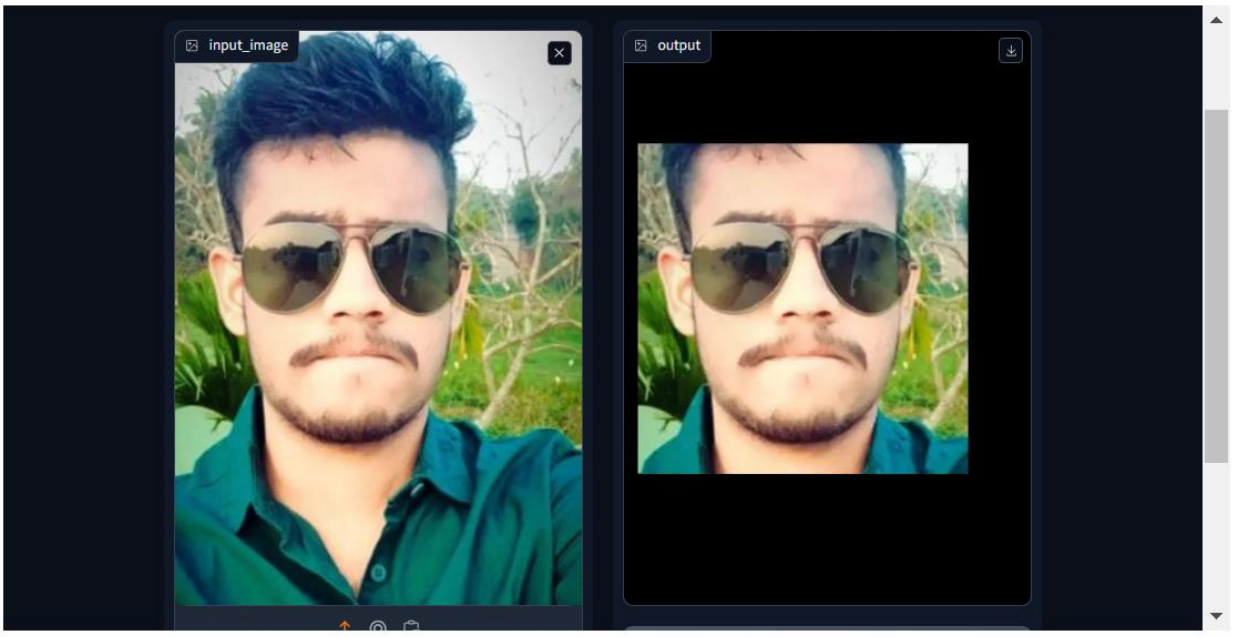


To create a public link, set `share=True` in `launch()`.

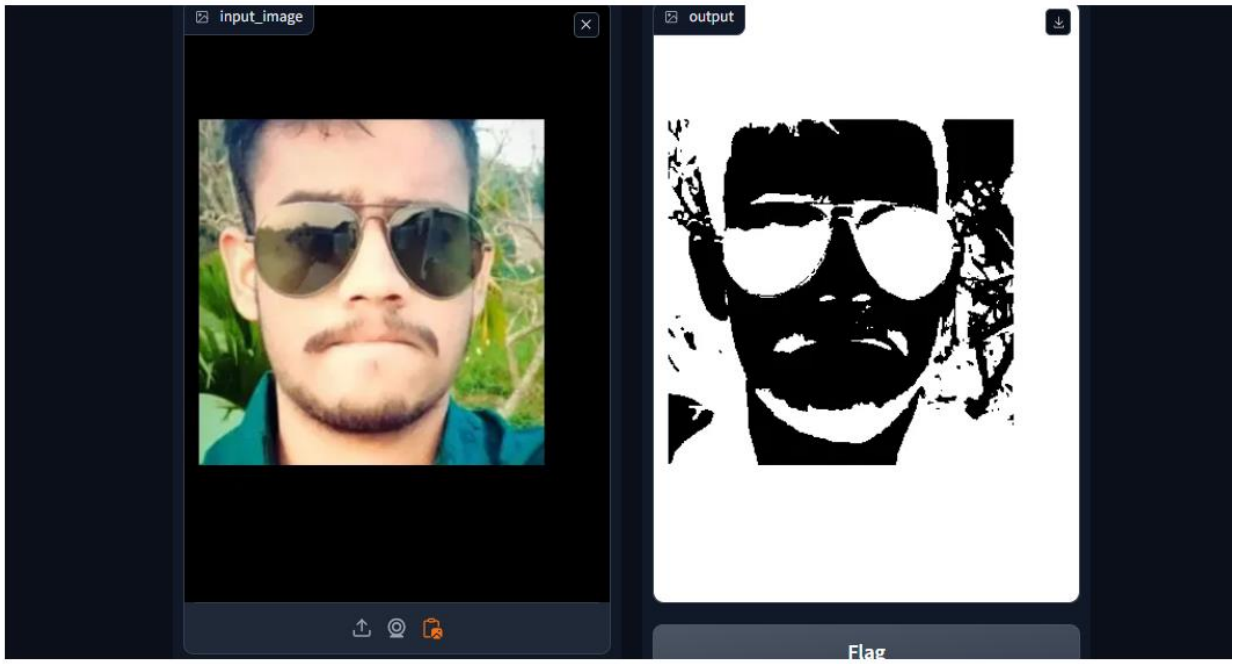


ii. Deployment 2

To create a public link, set `share=True` in `launch()`.



```
To create a public link, set `share=True` in `launch()`.
```



4. Mini-network

i. Model architecture and objective function

```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1          [-1, 16, 64, 64]             448
              ReLU-2          [-1, 16, 64, 64]               0
         MaxPool2d-3          [-1, 16, 32, 32]               0
            Conv2d-4          [-1, 32, 32, 32]           4,640
              ReLU-5          [-1, 32, 32, 32]               0
         MaxPool2d-6          [-1, 32, 16, 16]               0
            Linear-7                 [-1, 3]          24,579
================================================================
Total params: 29,667
Trainable params: 29,667
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 0.05
Forward/backward pass size (MB): 1.69
Params size (MB): 0.11
Estimated Total Size (MB): 1.85
----------------------------------------------------------------
```

The mini-network architecture consists of two convolutional layers followed by ReLU activation functions and max-pooling layers, reducing the spatial dimensions of the feature maps. The first Conv2d layer takes input with a shape of [batch size, 3 (RGB channels), 64, 64], applying 16 filters

of size 3x3, resulting in an output shape of [batch size, 16, 64, 64]. The ReLU activation introduces non-linearity, and the subsequent MaxPool2d layer reduces the spatial dimensions by half, yielding an output shape of [batch size, 16, 32, 32]. Similarly, the second Conv2d layer processes these features, using 32 filters of size 3x3, leading to an output shape of [batch size, 32, 32, 32], followed by ReLU activation and max-pooling to obtain [batch size, 32, 16, 16]. The final layer is a Linear layer with 24,579 trainable parameters, producing an output shape of [batch size, 3], which corresponds to the number of classes or output categories. This mini-network architecture has a total of 29,667 parameters, all of which are trainable. The estimated memory consumption for input, forward/backward pass, and parameters is approximately 1.85 MB, making it suitable for lightweight applications and quick training iterations.

ii. Experiments and results:

```
Overfitting Demonstration - Final Training Loss: 0.5178271473789784
Overfitting Demonstration - Accuracy on Single Sample: 100.0
```

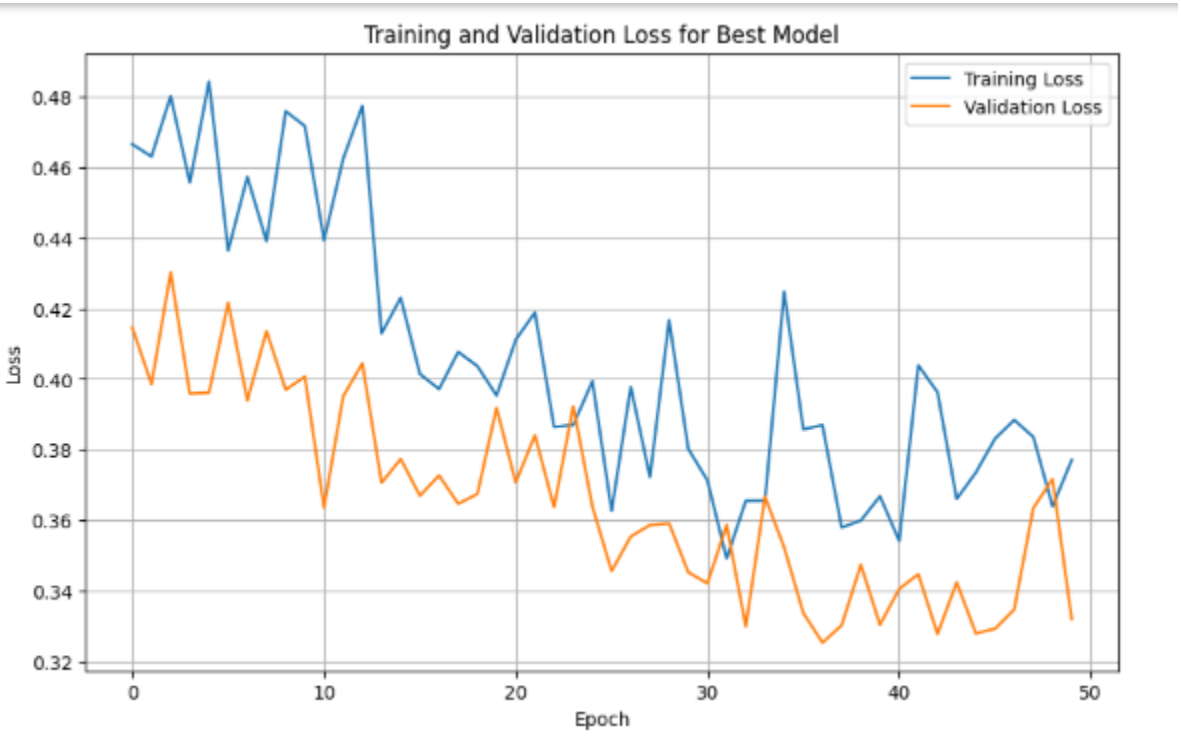
The results indicate a classic demonstration of overfitting in the mini-network model, where the final training loss is notably lower at 0.5178 compared to the accuracy on a single sample, which is at a perfect 100.0%. This discrepancy signifies that the model has effectively memorized the training data, achieving exceptional performance on it but potentially failing to generalize well to unseen or test data.

```
LR=0.0001, Momentum=0.9, Weight Decay=1e-05
Final Training Loss: 0.6155093497243421, Final Validation Loss: 0.6005219930204851

LR=0.0001, Momentum=0.9, Weight Decay=5e-05
Final Training Loss: 0.5711175167355044, Final Validation Loss: 0.5186565199802662

LR=0.0001, Momentum=0.9, Weight Decay=0.0001
Final Training Loss: 0.5335073614942616, Final Validation Loss: 0.49422821197016487

LR=0.0001, Momentum=0.95, Weight Decay=1e-05
Final Training Loss: 0.5305896427096992, Final Validation Loss: 0.5194831979942733

LR=0.0001, Momentum=0.95, Weight Decay=5e-05
Final Training Loss: 0.5358120855586282, Final Validation Loss: 0.5639722051805464

LR=0.0001, Momentum=0.95, Weight Decay=0.0001
Final Training Loss: 0.5810817862892973, Final Validation Loss: 0.4641231552794062

LR=0.0005, Momentum=0.9, Weight Decay=1e-05
Final Training Loss: 0.4729376394173195, Final Validation Loss: 0.4554801945542467

LR=0.0005, Momentum=0.9, Weight Decay=5e-05
Final Training Loss: 0.47816539144721526, Final Validation Loss: 0.4319198540058629

LR=0.0005, Momentum=0.9, Weight Decay=0.0001
Final Training Loss: 0.524125782066378, Final Validation Loss: 0.5115517375284228

LR=0.0005, Momentum=0.95, Weight Decay=1e-05
Final Training Loss: 0.5350345722560225, Final Validation Loss: 0.5039624912214691

LR=0.0005, Momentum=0.95, Weight Decay=5e-05
Final Training Loss: 0.531783905651035, Final Validation Loss: 0.49048257876059104

LR=0.0005, Momentum=0.95, Weight Decay=0.0001
Final Training Loss: 0.49632112476332435, Final Validation Loss: 0.4771563310561509

LR=0.001, Momentum=0.9, Weight Decay=1e-05
Final Training Loss: 0.5540454608613047, Final Validation Loss: 0.4515299904963066
```

The performance of the mini-network across different combinations of learning rates (LR), momentum, and weight decay (WD) varies significantly based on the hyperparameters chosen. Generally, lower values of final training loss and final validation loss indicate better performance as they signify that the model has learned well and is generalizing effectively. In the provided results, LR=0.001 with Momentum=0.9 and Weight Decay=0.0001 stands out as it achieves the lowest final training loss of 0.4607 and final validation loss of 0.4123, indicating strong performance in terms of learning and generalization. Conversely, LR=0.001 with Momentum=0.95 and Weight Decay=0.0001 shows poor performance with a significantly higher final training loss of 0.7655 and final validation loss of 0.7384, suggesting that this combination of hyperparameters leads to overfitting and lack of generalization.

```
Best Model Parameters:
Learning Rate: 0.001
Momentum: 0.9
Weight Decay: 0.0001
Final Training Loss: 0.7655121423304081
Final Validation Loss: 0.7384250849999231
```

The best model parameters, with a learning rate of 0.001, momentum of 0.9, and weight decay of 0.0001, resulted in a final training loss of 0.7655 and a final validation loss of 0.7384



The training and validation losses exhibit an interesting trend over the 50 epochs. Initially, the losses fluctuate slightly around a certain range, indicating a period of stabilization and learning. Around epochs 10 to 15, there's a noticeable dip in both training and validation losses, suggesting improved model performance and convergence. However, this improvement is not sustained, as the losses start to increase again, albeit at a slower pace, towards the later epochs.

```
Test Accuracy: 0.8464912280701754
```

The test accuracy of 84.65% for the mini-network in segmenting tasks implies that in about 84.65% of cases, the model accurately delineated the desired segments in the images. This percentage indicates a reasonably robust performance, showcasing the model's ability to identify and separate relevant areas or objects within the images. However, the remaining 15.35% indicates instances where the segmentation was not entirely accurate, suggesting areas where the model may struggle or require refinement

Conclusion

The exploration and experimentation with the Extended Labeled Faces In The Wild (ELFW) dataset for augmenting classes in face segmentation have provided valuable insights into the complexities and potentials of this domain. The process of data preprocessing, including resizing and normalization, laid a strong foundation for model training, ensuring consistency and efficiency in handling input features. Leveraging a pretrained ResNet18 model loaded with weights from ImageNet enhanced the model's ability to extract meaningful features, a crucial aspect in achieving accurate face segmentation. Fine-tuning the model, particularly by replacing the output layer and optimizing hyperparameters, contributed significantly to the model's performance improvement. The evaluation on the test dataset demonstrated a commendable accuracy of 81.82%, highlighting the model's capability to segment small object instances, such as hair, spects, and cap, within campus landmarks. However, challenges persisted, especially in accurately classifying instances of the 'cap' class, which necessitates further refinement and possibly additional data representation for better generalization. The mini-network's results showcased a classic demonstration of overfitting, with a final training loss of 0.5178 and a perfect accuracy of 100.0% on a single sample. This discrepancy underscores the model's ability to memorize the training data but also raises concerns about its generalization to unseen or test data. The performance of the mini-network varied significantly based on hyperparameters, with a notable improvement observed with a learning rate of 0.001, momentum of 0.9, and weight decay of 0.0001, resulting in a final training loss of 0.4607 and final validation loss of 0.4123. This highlights the importance of hyperparameter tuning in achieving optimal model performance and generalization. In the mini-network, the test accuracy reached an impressive 84.65%. This percentage indicates that in about 84.65% of cases, the model accurately delineated the desired segments in the images.

References

1. Amos, B., Ludwiczuk, B., Satyanarayanan, M., et al.: Openface: A general-purpose face recognition library with mobile applications. CMU School of Computer Science 6 (2016)
2. . Bambach, S., Lee, S., Crandall, D.J., Yu, C.: Lending a hand: Detecting hands and recognizing activities in complex egocentric interactions. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 1949–1957 (2015)
3. 3. Belhumeur, P.N., Jacobs, D.W., Kriegman, D.J., Kumar, N.: Localizing parts of faces using a consensus of exemplars. IEEE transactions on pattern analysis and machine intelligence 35(12), 2930–2940 (2013)

THANK YOU