Facial Expression Recognition

By: Ethan Assefa, Thomas Burrell, Tatev Gomtsyan

AGENDA

1 **MOTIVATION**
FER uses and goals

2 **OUR DATA**
Description of data

3 **MODELS**
Development process

4 **CONCLUSION**
Model results and comparison

MOTIVATION

Facial Emotion Recognition (FER) helps create more intuitive and human-like interactions between computers and people, leading to more effective and engaging communication.

Cross-industry applications (healthcare, security, marketing, etc.)

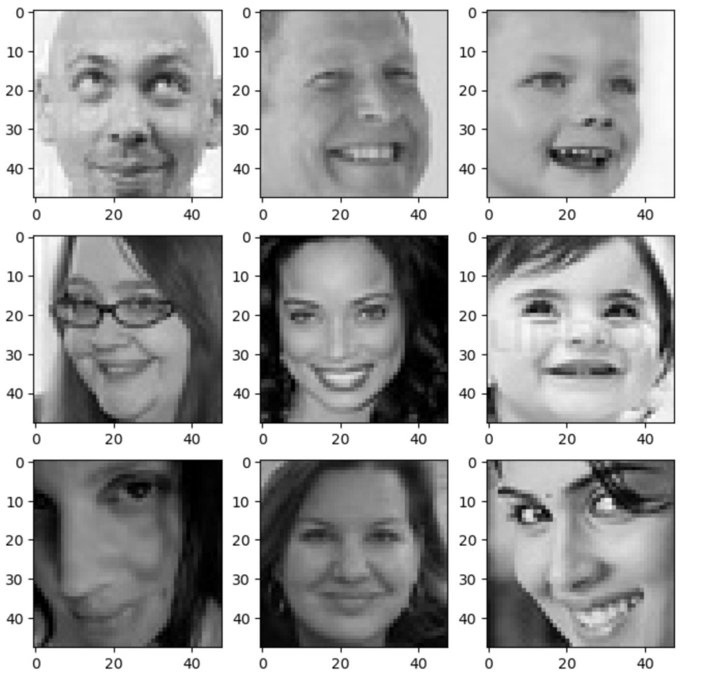
- Virtual assistants
- Customer service bots
- Social robots

Deep Learning allows for development of complex models that can recognize a wide range of expressions with high accuracy. CNNs are powerful pattern recognition tools in images. We explored a variety of techniques and models to discover the best approach to FER.

OUR DATA

Kaggle ‘face expression recognition’ dataset:

- ~29,000 images in Train folder
- ~7,000 images in Validation folder
- Seven expression categories
  - angry
  - disgust
  - fear
  - happy
  - neutral
  - sad
  - surprise



APPROACH: MODEL FROM SCRATCH

Model building	Parameters	Output
• Image preprocessing • 4 CNN Layers, flattened • 2 fully connected layers • Initialize Adam optimizer	• Learning rate: 0.001 (Adam adaptively adjusts based on gradients and previous updates during training) • Epochs: 20 • Batch size: 64	• Updated training and validation accuracies • Loss Value • Precision and Recall scores • Testing model on new images

TensorFlow/Keras: Model from Scratch

```
no_of_classes=7

model=Sequential()

#1st CNN layer
model.add(Conv2D(64,(3,3),padding="same",input_shape=(48,48,1)))
model.add(BatchNormalization())
model.add(Activation("relu"))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Dropout(0.25))


#2nd CNN layer
model.add(Conv2D(128,(5,5),padding="same"))
model.add(BatchNormalization())
model.add(Activation("relu"))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Dropout(0.25))

#3rd CNN layer
model.add(Conv2D(512,(3,3),padding="same"))
model.add(BatchNormalization())
model.add(Activation("relu"))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Dropout(0.25))

#4th CNN layer
model.add(Conv2D(512,(3,3), padding='same'))
model.add(BatchNormalization())
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))

model.add(Flatten())
```

Epoch 1

```
val_accuracy: 0.4037
```

```
Epoch 1/10
/Users/tatevgomtsyan/anaconda3/lib/python3.11/site-packages/keras/src/trainers/data_adapters/py_dataset_adapter.py:120: UserWarning: Your `PyDa
ould call `super().__init__(**kwargs)` in its constructor. `**kwargs` can include `workers`, `use_multiprocessing`, `max_queue_size`. Do not pa
nts to `fit()`, as they will be ignored.
  self._warn_if_super_not_called()
225/225 ━━━━━━━━━━━━━━━━━━━━ 851s 4s/step - accuracy: 0.2561 - loss: 1.9544 - val_accuracy: 0.4037 - val_loss: 1.6660 - learning_rate: 0.0010
```

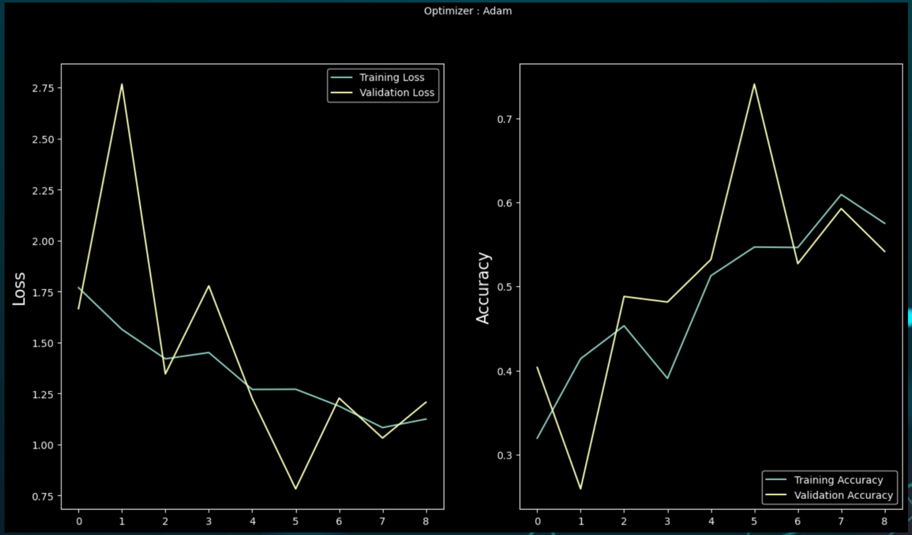
Epoch 9

```
val_accuracy: 0.5416
```

```
225/225 ━━━━━━━━━━━━━━━━━━━━ 0s 3s/step - accuracy: 0.5790 - loss: 1.1147
Epoch 9: ReduceLROnPlateau reducing learning rate to 0.00020000000949949026.
225/225 ━━━━━━━━━━━━━━━━━━━━ 757s 3s/step - accuracy: 0.5790 - loss: 1.1148 - val_accuracy: 0.5416 - val_loss: 1.2074 - learning_rate: 0.0010
Epoch 9: early stopping
Restoring model weights from the end of the best epoch: 6.
```

FEATURES

- 4 CNN layers, flattened
- Batch Normalization
- Relu activation
- Max pooling
- Dropout
- 2 fully connected layers



PyTorch: Pretrained Model

```python
class FaceModel2(nn.Module):
    def __init__(self):
        super(FaceModel2, self).__init__()
        self.eff_net = timm.create_model('resnet101', pretrained=True, num_classes=7)
        self.dropout = nn.Dropout(0.5)  # Dropout layer with 50% probability
        self.batch_norm = nn.BatchNorm1d(7)  # BatchNorm for 7 classes output from the model

    def forward(self, images, labels=None):
        logits = self.eff_net(images)
        logits = self.dropout(logits)
        logits = self.batch_norm(logits)

        if labels is not None:
            loss = nn.CrossEntropyLoss()(logits, labels)
            return logits, loss
        return logits
```

FEATURES

- Using pre-trained model from ‘timm’ (Torch Image Models) library
- Create model named FaceModel()
- ResNet-101 backbone
- pretrained=True (initialized weights from large dataset)

FINE-TUNING

- Data augmentation
- Epochs = 20
- Learn rate = 0.001
- Batch Size = 64

Epoch 1

```
EPOCHS[TRAIN]1/20: 100%|██████████| 451/451 [27:36<00:00,  3.67s/it, loss=1.493357, acc=0.415629, precision=0.415308, recall=0.415308]
EPOCH[VALID]1/20: 100%|██████████| 111/111 [02:43<00:00,  1.48s/it, loss=1.235303, acc=0.528721, precision=0.527805, recall=0.527805]
Saved Best Valid Loss
Saved Best Precision
Saved Best Recall
Epoch 1:
  Train - Loss: 1.493357, Acc: 0.415629, Precision: 0.415308, Recall: 0.415308
  Valid - Loss: 1.235303, Acc: 0.528721, Precision: 0.527805, Recall: 0.527805
```

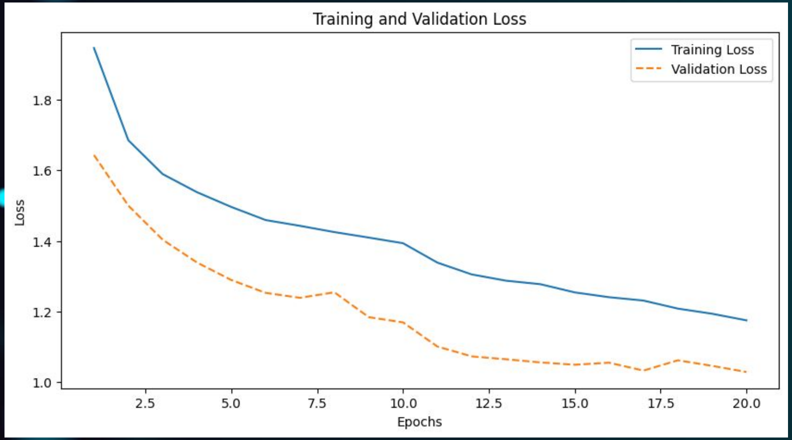
Acc: 0.528721,

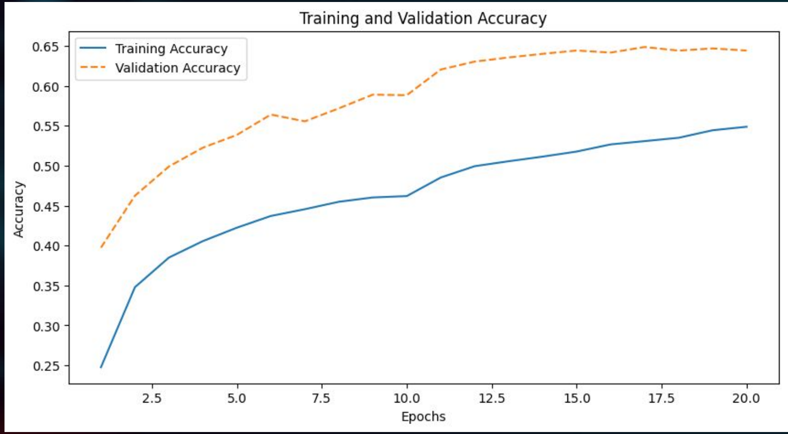
Epoch 20

```
EPOCH[TRAIN]20/20: 100%|██████████| 451/451 [00:20<00:00, 22.37it/s, acc=tensor(0.5489), loss=1.17]
EPOCH[VALID]20/20: 100%|██████████| 111/111 [00:02<00:00, 49.92it/s, acc=tensor(0.6443), loss=1.03]
Saved Best Model - Weights with Lowest Validation Loss
Epoch 20/20: Train Loss: 1.1747, Accuracy: 0.5489, Precision: 0.5526, Recall: 0.5526
Validation Loss: 1.0284, Accuracy: 0.6443, Precision: 0.5659, Recall: 0.5659
```

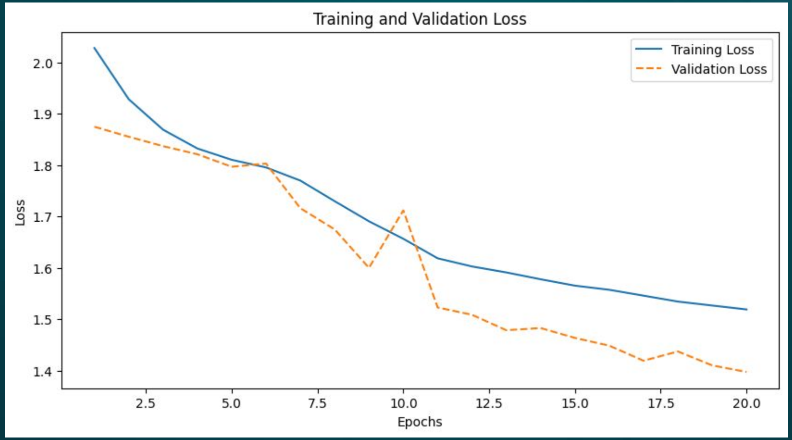
Accuracy: 0.6443,

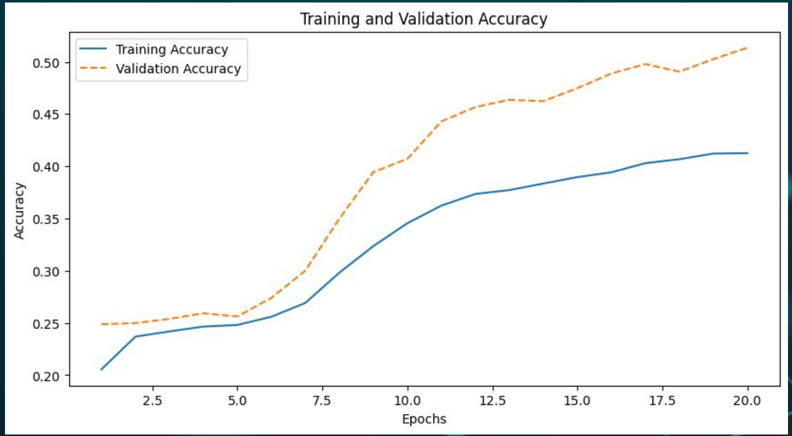
ResNet101 Results





EfficientNet (B5) Results





MODEL COMPARISON

Model Type	Accuracy
Individual Emotion Model	Angry - 86.41% Disgust - 98.43% Fear - 85.59% Happy - 74.17% Neutral - 82.79% Sad - 83.88% Surprise - 88.72%
Multi-class General Model (pyTorch):	42.15%
Ensemble Model (Single-Emotion and Multi-Class Models):	45.08%
Ensemble Model (Single-Emotion Models Only):	40.16%
Stacking:	40.83%
Resnet101 (WINNER!)	**64.43%**
EfficientNet (B5):	51.34%

MODEL TYPE PROS/CONS

Pre-trained	Model from Scratch
• Speeds up Development • Requires Less Data • Generalization Over Specialization • Limited Customization	• Highly customizable • Optimized for Specific Data • Requires Large Datasets • Time and Resource Intensive

CONCLUSION

CHALLENGES

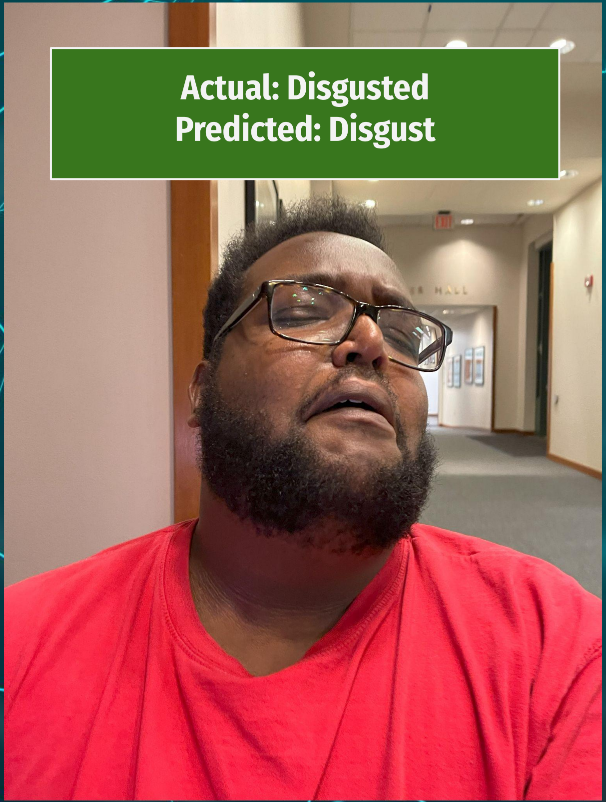
- Variability in emotional expressions, hard to create model that generalizes across diverse populations
- Cross-cultural differences, would need a more diverse dataset to ensure accuracy and inclusivity
- Privacy and ethical concerns, consent, potential misuse

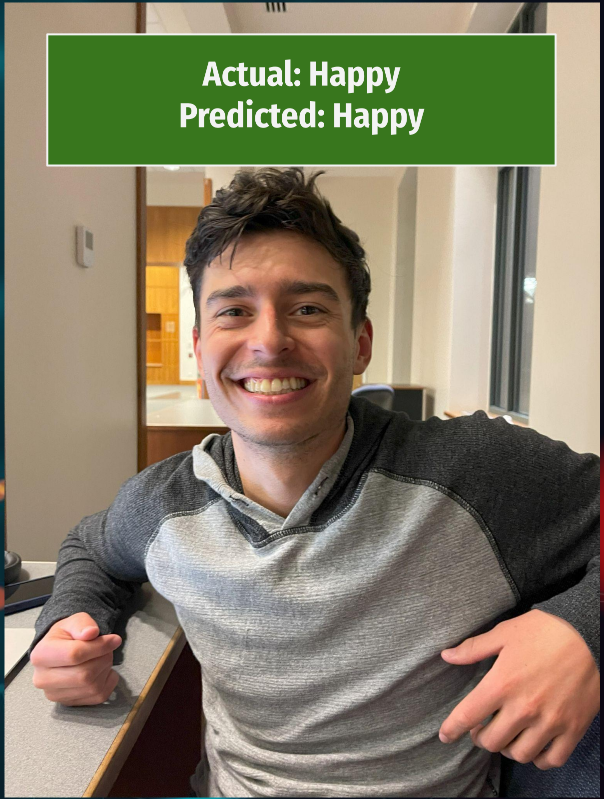
CONSIDERATIONS

- Make our own train/validation/test split from data
- Further fine-tuning parameters (more epochs, batch size)
- Better computational infrastructure (more resources)

FUTURE WORK:

- Real-time live video processing for mental health interventions (during telehealth or therapy sessions, with consent)
- Surveillance and security






```
ResNet Model: Image tatev_surprised.jpg is classified as Disgust
ResNet Model: Image ethan_disgust.jpeg is classified as Disgust
ResNet Model: Image trey_smile.jpeg is classified as Happy
```

THANK YOU!

Questions?