

Capstone Project: Facial Emotion Recognition



By Deepikaa Sriram

MIT Applied Data Science Program

Problem Definition



Deep learning is a method of processing data that is inspired by how humans process information



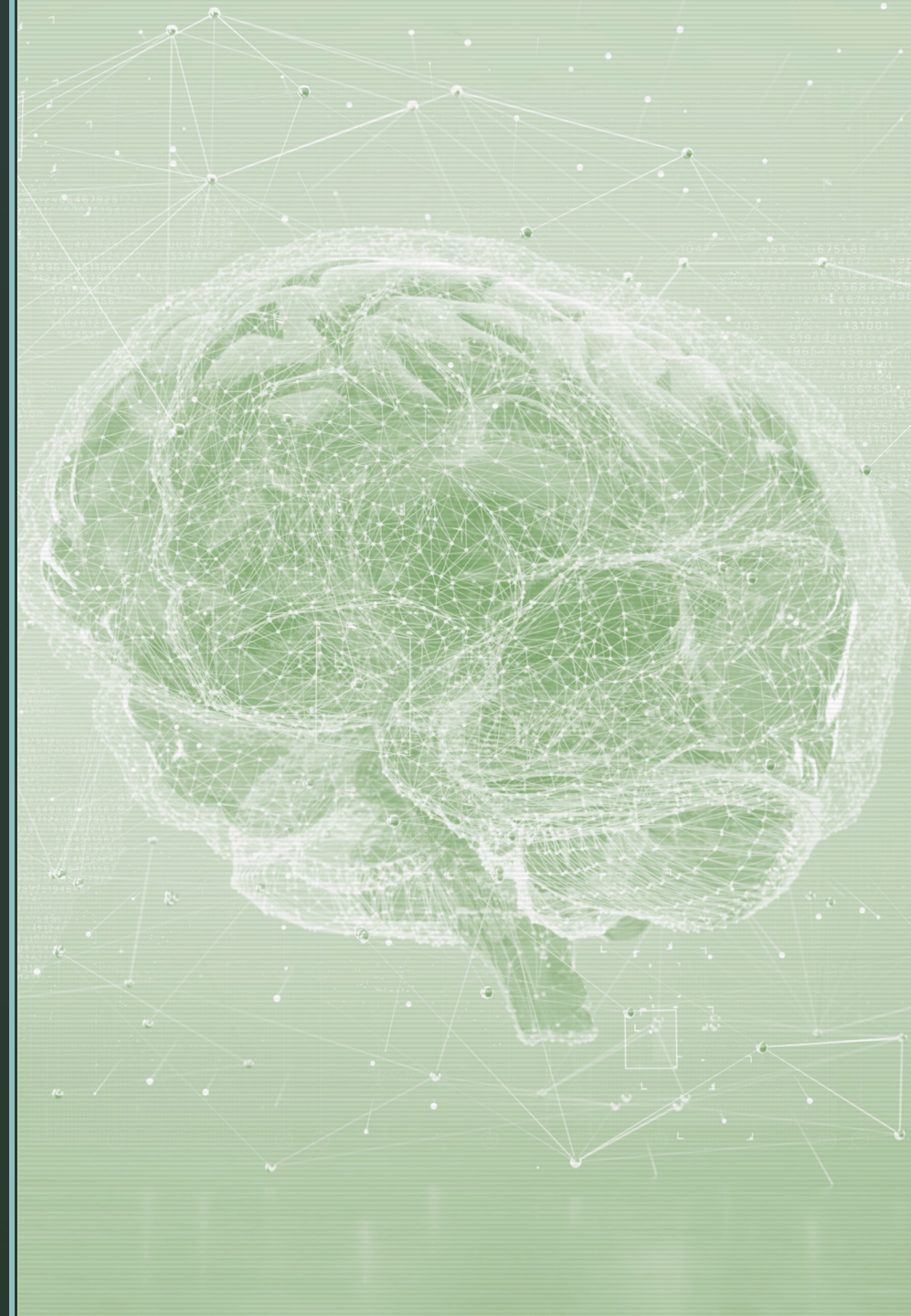
Deep Learning can recognize complex patterns in pictures, text, sounds, and other data to produce accurate insights and predictions.



Deep Learning has applications in many pattern-recognition and predictive tasks across all industries

Artificial Emotion Intelligence

- The study and development of technologies and computers
- Can read human emotions by means of analyzing body gestures, facial expressions, voice tone, etc. and react appropriately to them





Complexities with Facial Emotion Recognition

- Facial emotion recognition presents several complexities due to the variability:
 - Diversity in facial features, proportions, age, race, gender, cultural norms, etc.
 - Nuances in situational facial expressions, lighting conditions, etc.



Solution Approach

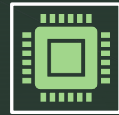
- The solution approach involves leveraging Deep Learning and AI techniques to train a computer vision model capable of accurately detecting facial emotions by performing the following steps:
 - Aggregating a large dataset
 - Labeling and pre-processing the data
 - Feature extraction using Convolutional Neural Networks (CNN)
 - Model training and validation
 - Evaluating the model using performance metrics
 - Fine-tuning and optimization

Industry Standards for Deep Learning Facial Recognition Models



Convolutional Neural Networks (CNNs)

CNNs have been extensively employed for FER tasks. They are well-suited for capturing spatial information in images.



VGG (Visual Geometry Group) Network

The VGG network is a deep CNN architecture known for its simplicity and effectiveness. VGG models with different depths, such as VGG16 and VGG19, have been employed for FER.



ResNet (Residual Network)

ResNet is a deep residual neural network architecture. Variants like ResNet-50, ResNet-101, and ResNet-152 have been used for FER tasks.



DenseNet

DenseNet facilitates feature propagation and encourages gradient flow, leading to improved performance.



MobileNet

MobileNet is a lightweight CNN architecture that allows for reasonable accuracy while reducing the computational cost.

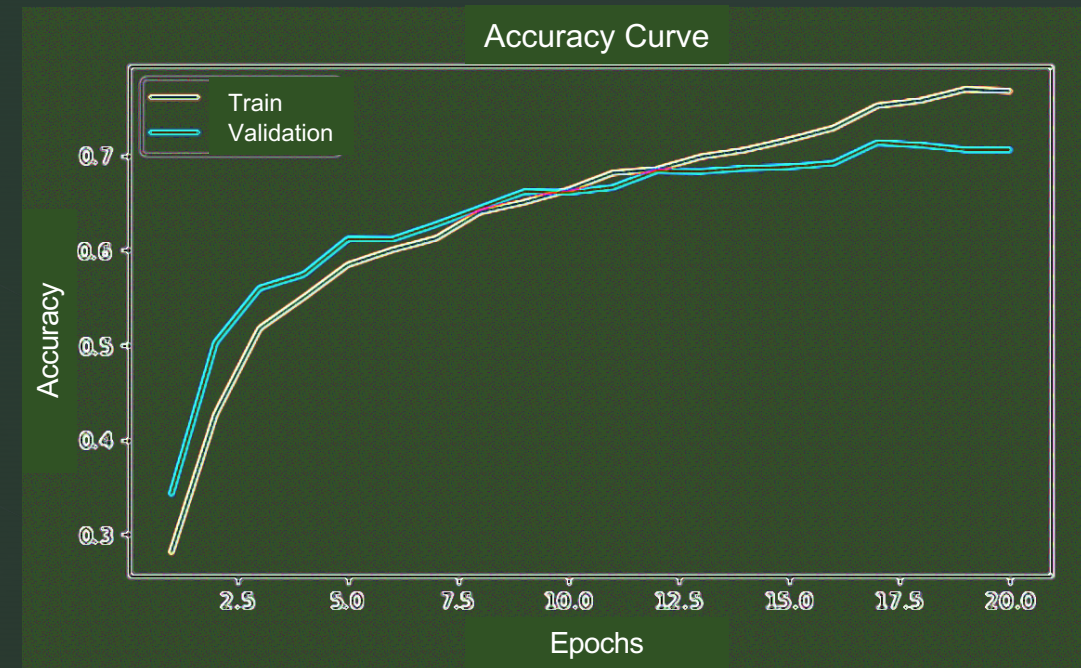
Comparative Model Performance

Accuracy on Test Dataset

CNN Model 1	CNN Model 2	VGG Model
75.7%	64.0%	60.9%
ResNet Model	EfficientNet Model	CNN Model 3
43.7%	25%	50.1%

Final Model Solution

- After training and fine-tuning the model on a large dataset of facial expressions, the final model achieved high accuracy in facial emotion recognition was a Connected Neural Network (CNN) Architecture



Use Cases for Facial Emotion Recognition



ENHANCING CUSTOMER
EXPERIENCE IN RETAIL
AND HOSPITALITY



MENTAL HEALTH
DIAGNOSIS



VIRTUAL ASSISTANT
PERSONALIZATION

Executing Business Solution

To successfully implement the facial emotion recognition solution in a business setting, the following steps should be taken:

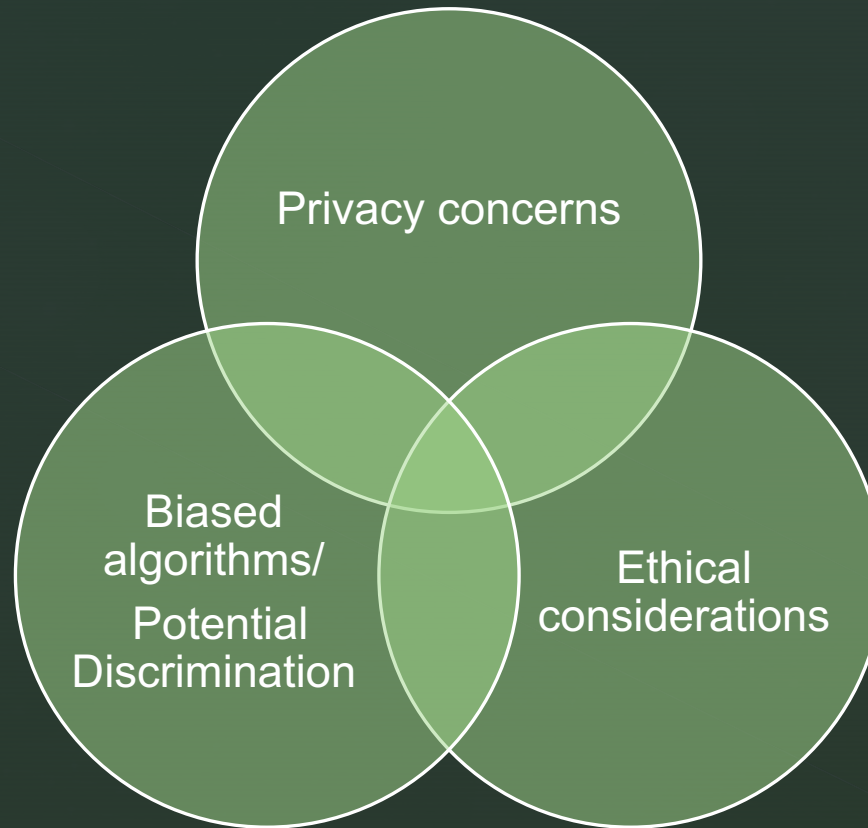
Dataset
collection

Model training

Integration
with existing
systems

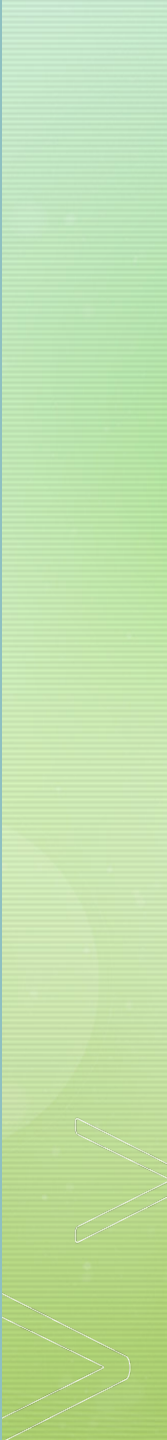
Continuous
monitoring and
improvement

Risks & Challenges

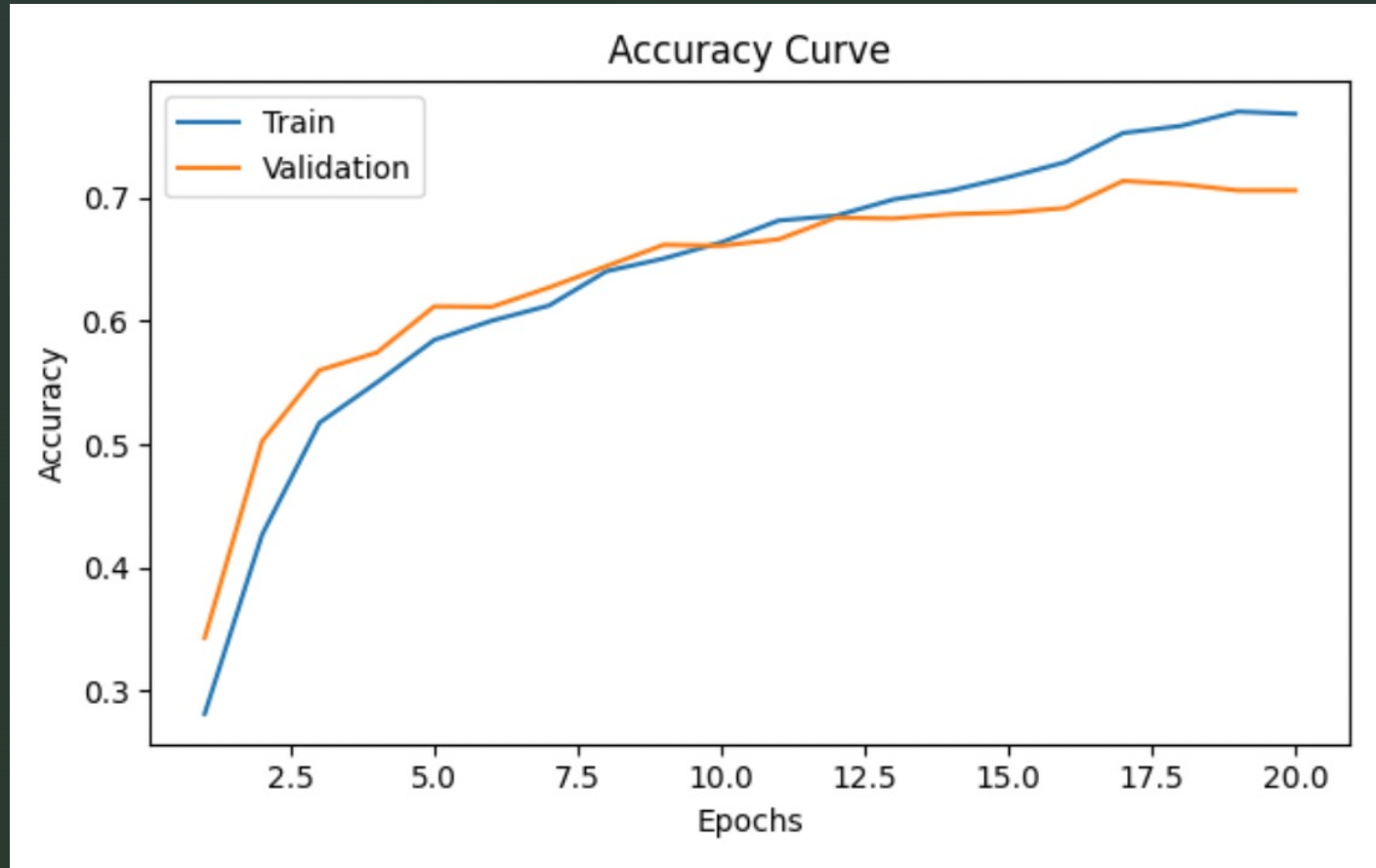




Executive Summary

- In summary, facial emotion recognition using Deep Learning and AI techniques offers significant potential for improving human-machine interaction and enabling emotionally intelligent systems
 - The process, albeit complex, holds tremendous potential to allow technology to more intuitively respond to human emotions
- 

Appendix A



Appendix B

```
# Create a Sequential model
model1 = Sequential()

# Add first Convolutional block
model1.add(Conv2D(filters = 16, kernel_size= (5,5), padding='same', activation='relu', input_shape=(224, 224, 1)))
model1.add(MaxPooling2D(pool_size=2))
#model1.add(Dropout(0.2))

# Add second Convolutional block
model1.add(Conv2D(filters = 32, kernel_size= (3,3), padding='same', activation='relu'))
model1.add(MaxPooling2D(pool_size=2))
#model1.add(Dropout(0.2))

# Add third Convolutional block
model1.add(Conv2D(filters = 64, kernel_size= (3,3), padding='same', activation='relu'))
model1.add(MaxPooling2D(pool_size=2))
#model1.add(Dropout(0.2))

# Add fourth Convolutional block
model1.add(Conv2D(filters = 128, kernel_size= (3,3), padding='same', activation='relu'))
model1.add(MaxPooling2D(pool_size=2))
#model1.add(Dropout(0.2))

# Add fifth Convolutional block
model1.add(Conv2D(filters = 256, kernel_size= (3,3), padding='same', activation='relu'))
model1.add(MaxPooling2D(pool_size=2))
#model1.add(Dropout(0.2))

# Add sixth Convolutional block
model1.add(Conv2D(filters = 512, kernel_size= (3,3), padding='same', activation='relu'))
model1.add(MaxPooling2D(pool_size=2))
#model1.add(Dropout(0.2))

# Flatten the input
model1.add(Flatten())

# Add first Dense layer
model1.add(Dense(512, activation='relu'))
model1.add(Dropout(0.4))

# Add final Dense layer
model1.add(Dense(4, activation='softmax'))

# Print model summary
model1.summary()
```


Appendix C

