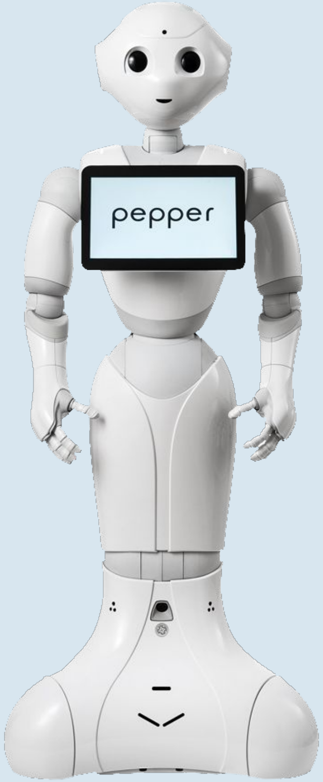
# Facial Emotion Recognition

Abdelouadoud Guelmami
S5467288



# Table of contents

01

## Aim

What's the goal we wanna reach

02

## How?

What are the main steps we need to take

03

## DataSet

Discussion of the dataSet

04

## Algorithms

What algorithms will we use?

05

## Results

Observation and Highlights

06

## Conclusion

What do we conclude?

# Aim

# Improve communication Skills for Pepper

## Recognition

Make Pepper able to recognize the emotions of the people

## Communication

Based on this emotion Pepper will be to choose more accurate phrases

## Improving

Over time Pepper will be to communicate more in a humanic way

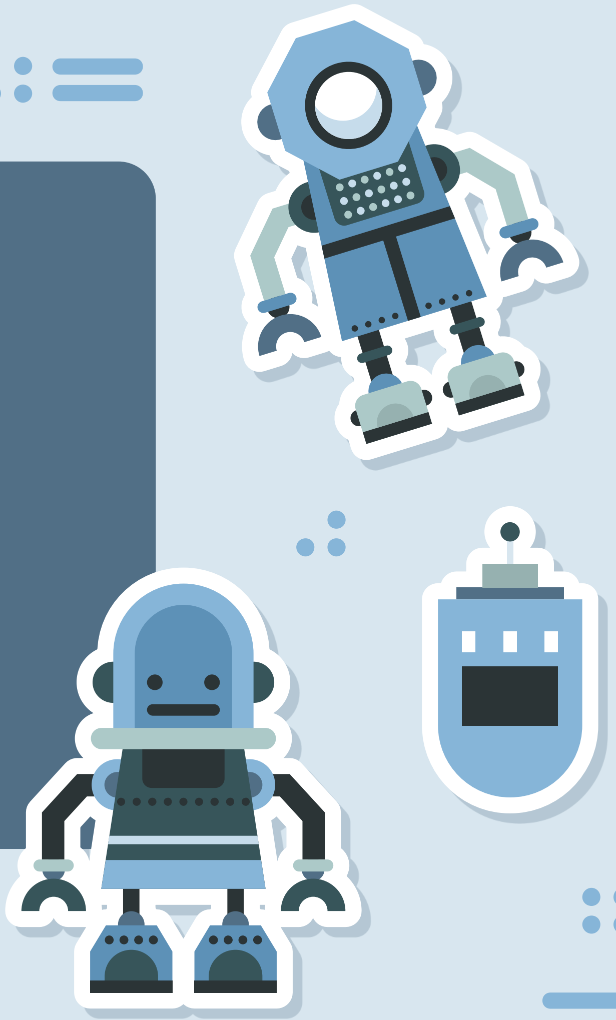
# How?

# Machine learning

- Ability to study and analyse large datasets
- Ability to provide high accuracy
- Continuous Improvements

Supervised machine learning

- CNN
- Random Forest
- SVM RBF



**DataSet:** https://www.kaggle.com/datasets/mhantor/facial-expression/data

**R. Forest:** https://colab.research.google.com/drive/1MK6233t68IpmljrVmoBePAcc2wulCrXJ?usp=sharing

**SVM RBF:** https://colab.research.google.com/drive/1vm3IX2QgQIndiUO5SQJPuELCzOlLrtZI?usp=sharing

**CNN:** https://colab.research.google.com/drive/1uC5aVR2lOBBA2WHe6aMrI4OSKK4qAYd1?usp=sharing

# DataSet

# DataSet

images.npy: Contains 19,950 images, size 48×48 pixels with RGB channels.

labels.npy: Contains one-hot encoded labels e.g [0, 0, 0, 1] indicates Surprised

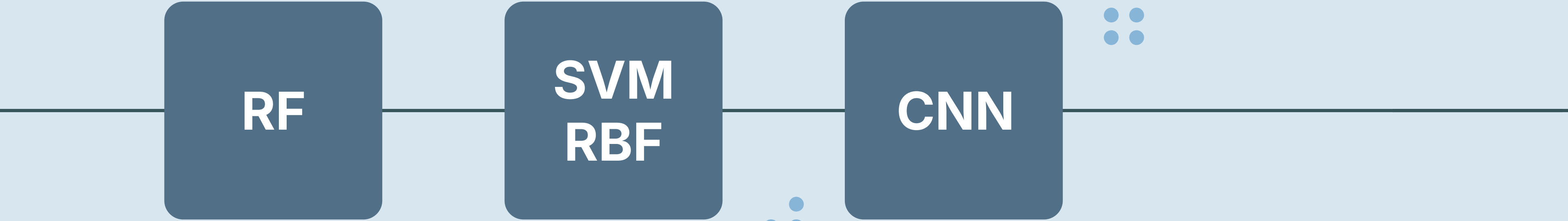
## Dataset Split:

| | | |
|---|---|---|
| Training: | 13,965 images | 70% |
| Validation: | 4,009 images | 20% |
| Testing: | 1,976 images | 10% |

| | |
|---|---|
| Happy | 7200 units |
| Angry | 3,850 units |
| Neutral | 5100 units |
| Surprised | 3800 units |

# Algorithms

# Models



RF

SVM RBF

CNN

An ensemble learning method using multiple decision trees to improve prediction accuracy by averaging their outputs.

Support Vector Machine model using the Radial Basis Function kernel to map data into higher dimensions for better class separation.

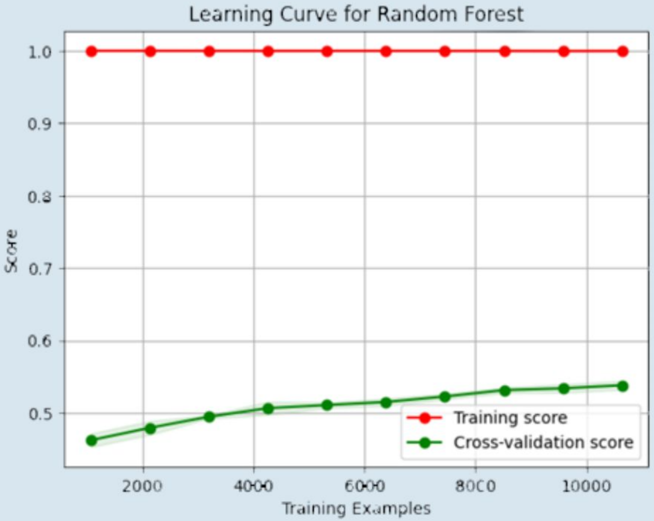
A deep learning model designed for image-based tasks, using convolutional layers to extract features and classify inputs effectively.

# Random Forest

- Best parameters found: max_depth:: 20 'max_features': 'sqrt' 'n_estimators': 200}

- Accuracy: 0.56

- Training process:
  - Trains multiple decision trees independently.
  - Combines predictions through voting



## Limitations:

- Overfitting: The near-perfect training accuracy coupled with poor test accuracy indicates overfitting, likely due to excessive model complexity (e.g., max_depth=20) or insufficient regularization.

- Generalization Issues: The model fails to generalize to unseen data, as evidenced by the wide gap between training and validation accuracy in the learning curve.

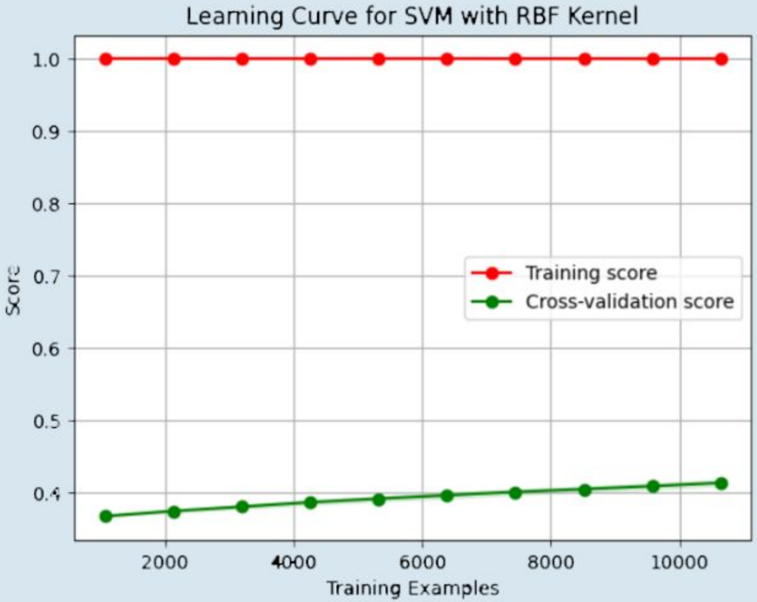
# SVM RBF

- Hyperparameters: C=10, gamma=0.001
- Accuracy: 0.42

Advantages:

- Handles non-linear relationships in the data well (provided sufficient tunin
- Works well on smaller datasets.

Limitations:
- Performance is heavily affected by class imbalance.
- Struggles with scalability and overfitting on large datasets.
- Requires significant effort to preprocess data and tune parameters.



# CNN

- Accuracy: 0.76

## Model Layers:

1st Block:

- **Conv2D:** 48 filters, kernel size (3x3), activation 'ReLU'
- **MaxPooling2D:** Pool size (2x2)
- **Dropout:** Rate 0.25

2nd Block:

- **Conv2D:** 64 filters, kernel size (3x3), activation 'ReLU'
- **MaxPooling2D:** Pool size (2x2)
- **Dropout:** Rate 0.25

3rd Block:

- **Conv2D:** 64 filters, kernel size (3x3), activation 'ReLU'
- **MaxPooling2D:** Pool size (2x2)

**Flatten Layer:** Converts output to 1024 units

## Dense Layers:

- **Dense:** 128 units, activation 'ReLU'
- **Dropout:** Rate 0.5
- **Dense:** 4 units (for 4 emotion classes), activation 'Softmax' (Output Layer)

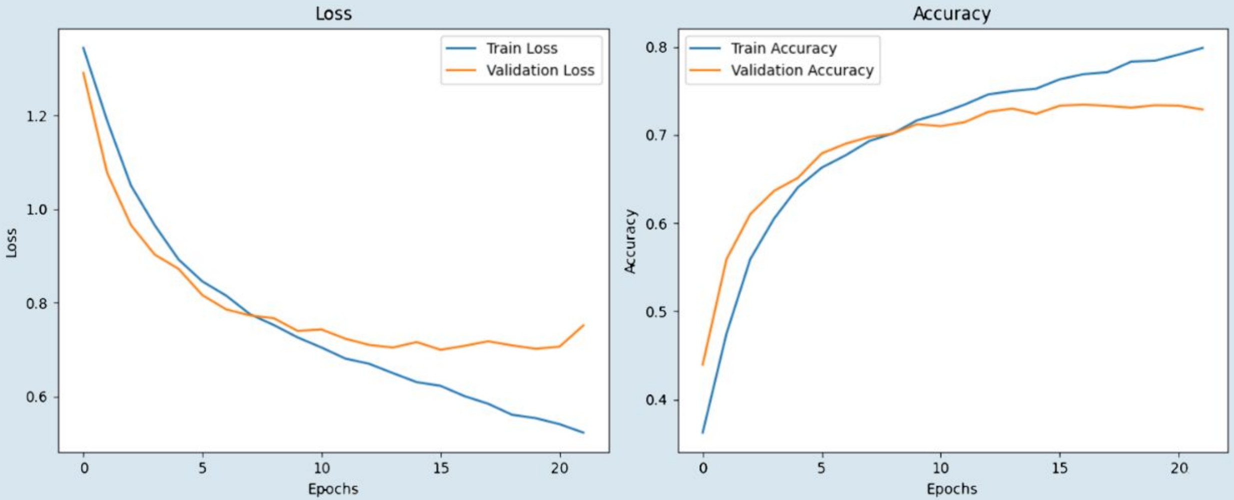
## Regularization:

- **Dropout:** Applied after Conv2D and Dense layers (to reduce overfitting)

## Model Compilation:

- **Optimizer:** Adam
- **Loss Function:** Sparse categorical cross entropy
- **Metrics:** Accuracy

# CNN


Loss
Train Loss
Validation Loss
1.2
1.0
0.8
0.6
Loss
0
5
10
15
20
Epochs
Accuracy
Train Accuracy
Validation Accuracy
0.8
0.7
0.6
0.5
0.4
Accuracy
0
5
10
15
20
Epochs

# CNN

**Advantages:**

Feature Extraction: Convolutional layers automatically extract spatial features from images, reducing the need for manual feature engineering.

Efficiency: Weight sharing in convolutional layers reduces the number of parameters compared to dense networks, making CNNs parameter-efficient.

End-to-End Learning: CNNs learn directly from raw image data, enabling robust performance with minimal preprocessing.

**Limitations:**

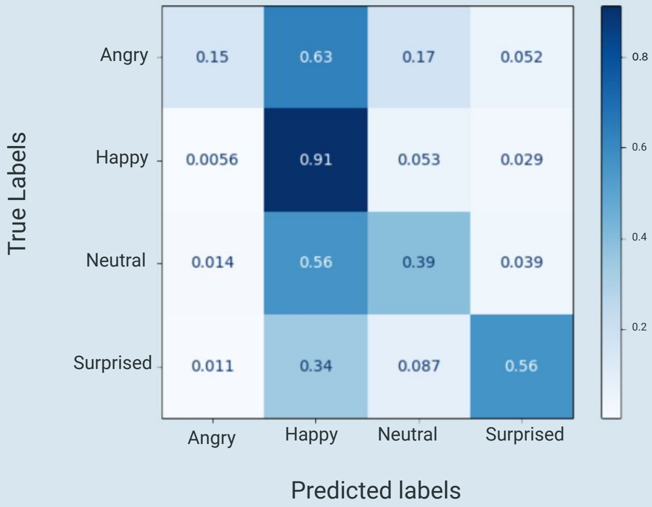
Class Imbalance: CNNs may struggle with underrepresented classes, leading to misclassifications (e.g., "Angry" confused with "Neutral").

Overfitting Risk: Without sufficient regularization, CNNs can overfit the training data, especially with limited datasets.

Architectural Improvements Needed: Incorporating advanced architectures like VGG or ResNet could enhance performance and reduce limitations.

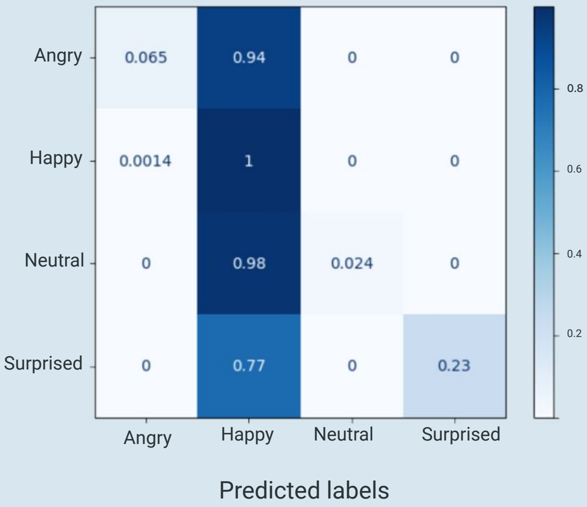
# Results

# Confusion Matrices



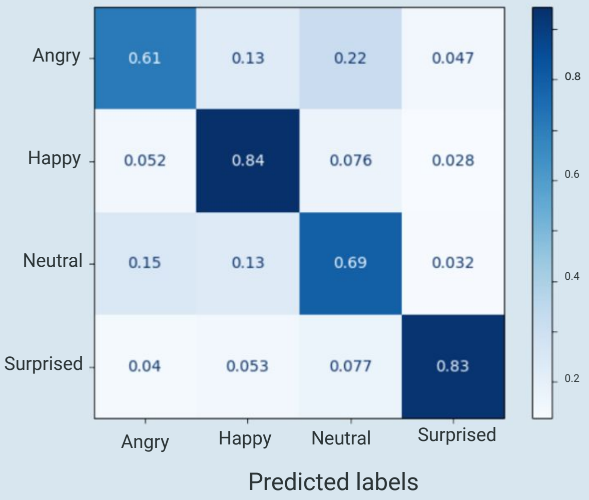
| True \ Predicted | Angry | Happy | Neutral | Surprised |
|---|---|---|---|---|
| Angry | 0.15 | 0.63 | 0.17 | 0.052 |
| Happy | 0.0056 | 0.91 | 0.053 | 0.029 |
| Neutral | 0.014 | 0.56 | 0.39 | 0.039 |
| Surprised | 0.011 | 0.34 | 0.087 | 0.56 |

**CNN**



| True \ Predicted | Angry | Happy | Neutral | Surprised |
|---|---|---|---|---|
| Angry | 0.065 | 0.94 | 0 | 0 |
| Happy | 0.0014 | 1 | 0 | 0 |
| Neutral | 0 | 0.98 | 0.024 | 0 |
| Surprised | 0 | 0.77 | 0 | 0.23 |

**SVM RBF**



| True \ Predicted | Angry | Happy | Neutral | Surprised |
|---|---|---|---|---|
| Angry | 0.61 | 0.13 | 0.22 | 0.047 |
| Happy | 0.052 | 0.84 | 0.076 | 0.028 |
| Neutral | 0.15 | 0.13 | 0.69 | 0.032 |
| Surprised | 0.04 | 0.053 | 0.077 | 0.83 |

**R. Forest**

# Classifications Report:

| Comparison of Models | CNN | Random Forest | SVM RBF |
|---|---|---|---|
| Test accuracy | 75% | 56% | 42% |
| Best Class (F1 Score) | Surprised (~84%) | Happy (~65%) | Happy(~56%) |
| Worst Class (F1 Score) | Angry(~63%) | Angry(~25%) | Neutral(~5%) |
| Overfitting | Low | High | Moderate |
| Computation | High | Low | High |
| Generalization | Excellent | Poor | Moderate |
| Observation | Balanced Performance, minimum confusion across classes | Moderate performance but struggles with “angry” and “Neutral” misclassification. | Overwhelming bias towards happy, fails to distinguish among other classes. |

# Observations:

## CNN:

Shows strong performance overall with some misclassifications between adjacent emotions like "Angry" and "Neutral," which is expected due to the subtle distinctions in expressions.

## Random Forest:

SVM fails to distinguish classes other than "Happy." This indicates that it heavily biases predictions towards the most frequent or dominant class, likely due to imbalance or limitations in kernel adaptability for this task.

## SVM RBF:

Random Forest struggles with class imbalance and fails to separate subtle differences between emotions like **Angry** and **Neutral**, leading to high confusion with dominant classes like **Happy**.

# Observations:

1.CNN:

**Strengths:** Achieves the highest validation and test accuracy.
**Performance:** Excellent generalization and balanced performance across classes.

2. Random Forest

**Strengths**: Performs moderately well for the "Happy" class.
**Weaknesses:** Performs poorly for other classes, especially "Angry."
**Concerns:** Exhibits high overfitting issues.

3. Support Vector Machine (SVM):

**Strengths:** Moderate success for the "Happy" class.
**Weaknesses:** Struggles with generalization and class balance, particularly for the "Neutral"

# Conclusion

# Conclusion

CNN is the most effective model for emotion detection, offering high accuracy, balanced predictions, and strong generalization. Random Forest and SVM are less suitable due to class imbalance and poor performance on subtle distinctions. CNN is the best choice for this task.

# Thank You