Human emotional state recognition based on facial expression using deep learning

Carl Emil Hejslet[1] **and Nicky Hauerbach Vetterstein**[2]

[1]**chejsl23@student.aau.dk**
[2]**nvette20@student.aau.dk**

1 INTRODUCTION

Facial emotion recognition (FER) is an important computer vision problem, which aims to classify human emotions. The model does so by using pictures of facial emotions as input, and categorizing the emotions into seperate classes (e.g. happy, angry). This is a classic computer vision task, and it has several real world applications like behavourial monitoring and entertainment purposes.
The task however is difficult, as many factors make it hard to decipher the correct emotion. Lighting differences, different poses, shadows and occlusion. Some emotions are also very similar visually speaking, like fear and disgust. All these things makes it more difficult to classify in practice.
Furthermore, The dataset used contains low resolution images (40×40px), and contains imbalance, which demands preprocessing.
For this kind of image recognition computer vision problem, CNNs have proven to be very effective. They learn features automatically, which means there is no need for handmade feature engineering, making it the perfect match for the dataset - FER2013 and FER problems in general.
The scope of the project is to:

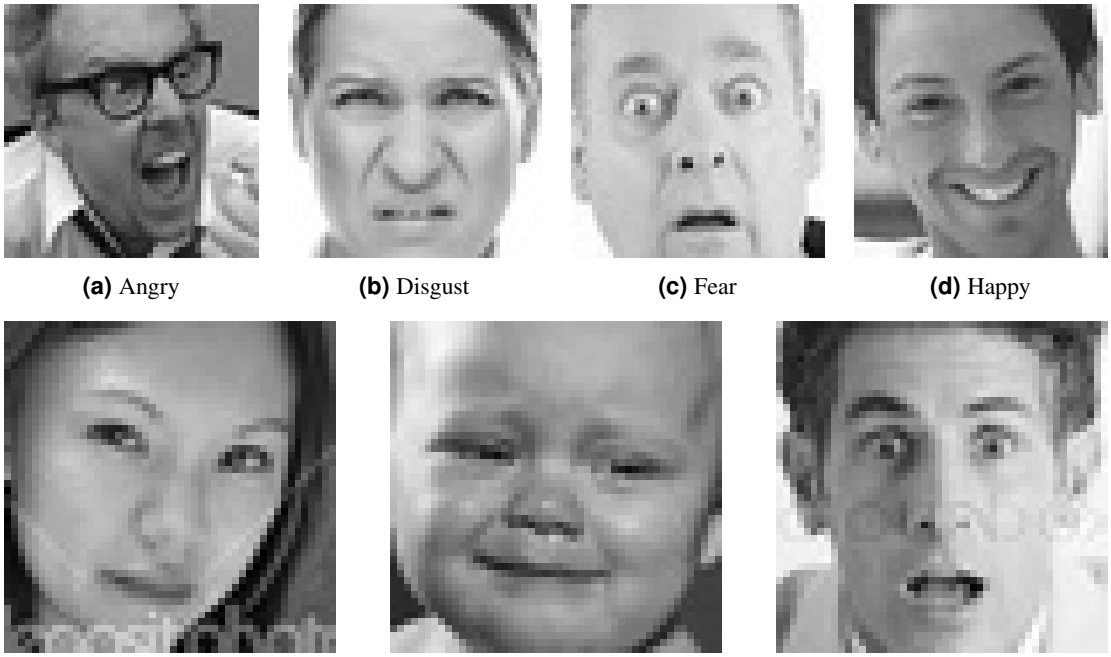
- Develop af preprocessing pipeline for augmentation and balancing
- Design and train a CNN model.
- Evaluate the performance of the model on emotion recognition.
- Understand training dynamics and model behaviour, regarding overfitting and generalization.

The remainder of the report is structured as follows: Section 2 presents the dataset, Section 3 describes the preprocessing steps, Section 4 details the model architecture, Section 5 outlines the training procedure, and Section 6 presents the experimental results. Section 7 concludes the report with a discussion of the findings and potential improvements.

2 DATASET

The dataset chosen is the image dataset FER2013 from kaggle. It consists of 7 different folders, containing grayscale images of classic human emotions: anger, disgust, fear, happy, neutral, sad, surprise. FER2013 originates from the ICML 2013 Workshop on Challenges in Representation Learning. SAMBARE (2020)

This makes the dataset perfectly applicable for a facial emotion recognition task. The images are 48x48 pixels in size, and are all closeups of faces, in order to give a model the right features.
Example image of each emotion is seen below:



(a) Angry **(b)** Disgust **(c)** Fear **(d)** Happy

(e) Neutral **(f)** Sad **(g)** Surprise

Figure 1. Example images for each emotion in the dataset.

Each emotion image has been gathered by scraping google image searches, using keywords related to the given emotion. This is a quick approach to collecting a visual emotion dataset, however, this approach also leads to imbalances in the dataset, as there are way more images of people being happy on the internet, than there are images of people being e.g disgusted. The exact differences are listed in table 1:

Emotion	Train magnitude	Test magnitude
Angry	3995	958
Disgust	436	111
Fear	4097	1024
Happy	7215	1774
Neutral	4965	1233
Sad	4830	1247
Surprise	3171	831

Table 1. Image count for each emotion in the train and test sets.

As seen on 1, the dataset is heavily unbalanced. In order to get useful and wellperforming model, it is essential to do some preprocessing to the dataset, before training a model.

3 PREPROCESING

A more balanced dataset would create less bias and underfitting in relation to the smaller classes in the dataset. This could be solved by creating synthetic augmented data, where small changes are being made to the existing pictures, to get more data, without changing the overall facial emotion. With this in mind, a data augmentation pipeline has been made:

Augmentation Pipeline

1. **Minor rotations** To slightly augment the image without losing semantic meaning, there has only been made small rotations to the original images. Therefore, in this case the image will only be rotated with a maximum of 15°. This makes the model more robust against images where the face is not completely centered.

```
angle = np.random.uniform(-15, 15)
M = cv2.getRotationMatrix2D((w/2, h/2), angle, 1.0)
out = cv2.warpAffine(out, M, (w, h), flags=cv2.INTER_LINEAR,
borderMode=cv2.BORDER_REFLECT_101)
```

2. **Horisontal flip** A horisontal flip was applied with a 50% probability, to enhance robustness against left/right assymmetry in facial expressions. Mostly, emotional facial features are invariant to horizontal orientation, which makes this transformation increase data diversity, without changing the semantic meaning of the image.

```
if np.random.rand() <0.5:
out = cv2.flip(out, 1)
```

3. **Zoom** A raondom zoom in the range [0.95, 1.08] was applied to simulate variations in camera distance and face scale. This encourages the model to become invariant to small changes in facial size and framing.

```
zoom = np.random.uniform(0.95, 1.08)
nw, nh = int(w*zoom), int(h*zoom)
out2 = cv2.resize(out, (nw, nh), interpolation=cv2.INTER_LINEAR)
```

4. **Brightness/contrast jitter** To simulate realistic lighting variations, each image was randomly adjusted in brightness and contrast. The pixel intensities were linearly transformed by α and β, where $\alpha \in [0.9, 1.1]$ controls contrast and $\beta \in [-8, 8]$ shifts brightness. This encourages the model to generalize better under different lighting conditions and camera exposures.

```
alpha = np.random.uniform(0.9, 1.1)
beta = np.random.uniform(-8, 8)
```

5. **Gaussian noise** With a 30% probability, weak Gaussian noise ($\mu = 0, \sigma = 3$) was added to the image to sumilate sensor noise and compression artefacts. This helps the model become more robust to minorpixel-level variations and prevents overfitting to overly clean training data.

```
if np.random.rand() < 0.3:
noise = np.random.normal(0, 3, out.shape).astype(np.float32)
```

The image augmentation methods were inspired by: Singh (2020)

Balancing the train folder

To make sure the class distribution in the training data is more even, a separate balancing script was implemented. The function `balance_train_folder` takes a source directory, and a destination directory, where each class is stored in its own subfolder (e.g. angry, happy).

For each emotion class, all image paths are collected and the number of samples is compared to a predefined target (in this case 5000 images per class):

- **Undersampling of majority classes:** If a class contains more than the target number of images, a random subset of exactly `target` images is selected using `random.sample`, and only these images are copied to the balanced training folder. This prevents classes with many examples from domination the loss and driving the decision boundaries.
- **Oversampling of minority classes:** If a class contains fewer than `target` images, all original images are first copied to the destination. Then, additional samples are generated until the class reaches the target size. Each extra image is produced by reading a randomly chosen source image and passing it through the `augment_face` pipeline described above. The augmented image is then saved with a unique filename using `uuid4` to avoid collisions.

This procedure makes a new training set where each emotion class has the same number of samples, combining random undersampling of majority classes and augmentation-based oversampling of minority classes. As a result, the model is trained on a class-balanced dataset, which reduces bias towards frequent emotions and allows the classifier to learn more informative decision boundaries for underrepresented expressions.

4 MODEL ARCHITECTURE

4.1 Model Type

The model chosen for the problem is a convolutional neural network (CNN). It is a specialized type of deep learning architecture designed for processing grid-like data such as images. CNNs apply learnable convolutional filters that slide over an input image and extract local spatial features, such as edges, corners, textures, and higher-level structures. Through stacked convolutional layers, the network gradually builds a hierarchical representation, where early layers capture simple visual patterns and deeper layers capture increasingly abstract features.

Compared to a fully connected network, a CNN contains far fewer parameters due to weight sharing in the convolutional kernels. THis make CNNs computationally efficient and much better suited for visual tasks. Pooling layers reduce spatial resolution and provide translation invariance, while dropout layers help prevent overfitting. Finally, fully connected layers map the learned feature representations to the final emotion classes using a softmax output layer.

In summary, a CNN is an appropiate and effective architecture for the project. Due to the fact that it can automatically learn relevant visual features directly from the pixels, without the need of handcrafted feature extraction. Kromydas (2023)

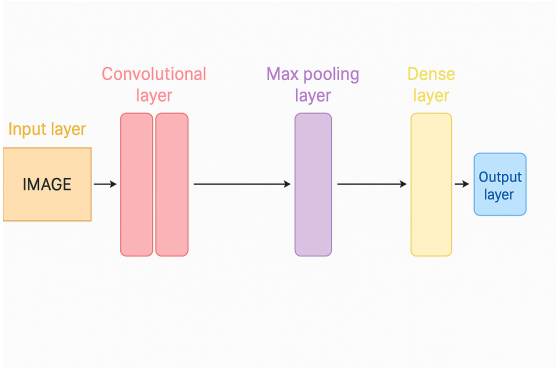


Figure 2. Illustration of a simple CNN architecture

4.2 CNN Relevancy For Facial Emotion Recognition

Facial emotion recognition (FER) is fundamentally a visual pattern recognition task, where subtle differences in facial regions such as the eyebrows, eyes, mouth shape, and wrinkles encode the underlying emotional state. CNNs are particularly well suited for FER because they excel at extracting such local and spatially coherent patterns.

By convolving small filters over the facial image, a CNN can learn to detect emotion-relevant features such as raised eyebrows (surprise), tightened lips (anger), or downturned mouth corners (sadness). These

features are often localized and scale-invariant, which alligns naturally with the convolutional operation. Furthermore, the hierarchical structure of a CNN allows deeper layers to combine low level patterns into more complex representations such as facial muscle configurations or expression components.

FER-2013 consists of low resolution, grayscale images (48×48), making traditional simple feature methods less effective. CNNs, on the other hand, can learn robust representations even from noisy or low quality input data. Their ability to generalize across variations in lighting, orientation, and slight pose differences futher strenghtens their suitabiity for emotion recognition.

Therefore CNNs provide a powerful and compared to state of the art - strong approach for FER, balancing accuracy, computational efficiency, and robustness to these images containing real world variability.

4.3 Our CNN architecture

Based on the principles of Convolutional Neural Networks as described above, a sequential architecture aimed at fitting the FER-2013 dataset was designed. The model is constructed using the Keras library and it consists of three main convolutional network blocks followed by fully connected layers.

The convolutional blocks The network takes a *48 x 48* pixel greyscale image as input. It has been structured in such a way that the model increases the number of filters while reducing the spatial dimensions of the image from block to block:

1. **Block 1:** Consists of two Conv2D layers with 64 filters each.
2. **Block 2:** The depth is doubled to two Conv2D layers of 128 filters each.
3. **Block 3:** Again the depth is doubled to one Conv2D layer with 256 filters.

Each convolutional layers uses a *3 x 3* kernel size and the ReLU activation function to introduce non-linearity. Aditionally, batch normalization was added after each convolutional layer, as this could help stabilize the learning process and allow the model to train faster by normalizing the inputs internally between the layers.

To reduce the size of the feature maps and control the computational costs, application of max pooling (*2 x 2*) at the end of each network block is favourable.
At the end of each network block, there is added dropout too, as deep learning models can easily overfit on small to moderately sized datasets like FER-2013. Each convolutional block therefore integrates a dropout rate of 0.25 with a stronger dropout rate of 0.5 being applied to the fully connected layers. This helps in forcing the network to learn more robust features by simply randomly dropping (turning off) a fraction of neurons during training, so that the network will not rely too heavily on any single pattern.

Finally the 2D feature maps are flattened into a 1D vector and passed through a dense layer with 512 neurons. The output layer is made of 7 neurons with a softmax activation function providing a probability distribution across the seven emotion classes (e.g. Happy, Sad, etc.).

5 TRAINING AND EVALUATION

5.1 Training setup

The model was trained for a maximum number of epochs of 50 with a batch size of 64. Then the Adam optimizer was used, with an initial learning rate set to 0.001. Adam was chosen, because it adaptively adjusts the learning rate for each parameer during training, which generally gives better convergence for image classification tasks than standard Stochastic Gradient Descent (SGD).

For the loss function, categorical crossentropy which is a standard choice for multi-class classification when the target labels are one-hot encoded was chosen.

5.2 Data preprocessing

Before feeding the images into the network, application of a custom standardization step that will center the data around zero was necessary. This can help the gradient descent algorithm converge faster and in finding a better minimum.

Instead of simply scaling pixel values into the range of *[0, 1]*, normalization of the data based on specific statistics of the training data was done. Then calculation of the mean and standard deviation in pixel values for images in the FER-2013 dataset.

5.3 Training and callback

The model uses several callbacks during training in order to prevent overfitting and to ensure that the training process runs efficiently:

1. **ModelCheckpoint:** This saves the model weights when the validation accuracy has improved. This makes it so that if the model begins to overfit towards the end of training (the 50 epochs), it still saves only the best performing model.
2. **EarlyStopping:** Monitoring of the validation loss with a patience of 10 epochs was added. If the model does not improve for 10 epochs training is stopped early. This saves on computational ressources by preventing the model from churning through all 50 epochs if a minimum is likely to have been found.
3. **ReduceLROnPlateau:** If the validation loss plateaus for 5 epochs the learning rate is reduced by a factor of 0.2. This can help in allowing the model to take smaller and finer steps to fine-tune the weights when it gets stuck in a local minimum.

5.4 Evaluation metrics

The performance of the model is largely based on the accuracy (number of correctly classified emotions) and loss. During training there are plots of the accuracy and the loss curves for both training and validation sets. Then visually, look for the point during training where the training loss continually decreases while the validation loss is starting to rise, as this will indicate a tendency towards overfitting to the training set.

By using the callback functions described above, the aim is to stop training before this effect becomes too pronounced.

6 RESULTS

6.1 Quantitative results

Our primary metric for meassuring how succesful the trained model was performing was the validation accuracy. We did two experimental phases to improve the initial performance of our model.

Phase 1: Baseline overfitting model Our initial larger model architecture achieved a very high training accuracy of approximately 90-93%, however it plateaued at a validation accuracy of 63%. The loss curves also showed large divergence which indicated that the model was very likely overfitting to the training data (i.e. learning the training data by heart instead of learning patterns that would generalize well).

Phase 2: Lighter architecture To try and combat this we drastically reduced the model architecture complexity, reducing the number of parameter down from +5 million to 1.5 million. We also increased the dropout rate from 25% to 40%, meaning the model was less likely to contain enough complexity that it could learn very specific patterns from the dataset.

As seen in 4 this improved the performance of the model - not by performing better on new unseen data (i.e. the validation set), but by closing the gap and stabilizing the performance metric curves between the train and validation sets. The loss curves observed in 4 align much more closely throughput the length of training on the revised CNN-architecture. While the final validation accuracy did not improve, but hovered at 64%, the model is now more robust and less prone to overfitting.

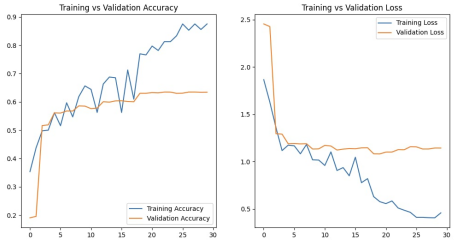


Figure 3. Training and validation for the initial model architecture. Here there is a significant gap (of roughly 30%) between the accuracy scores for the train and validation sets along with strong divergence in loss functions, indicating significant overfit from the train data.

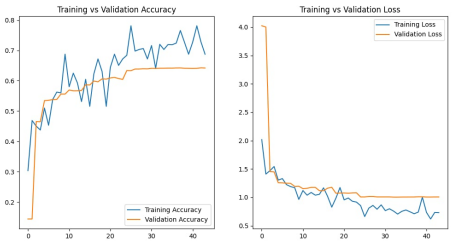


Figure 4. Training and validation metrics for the final model architecture of the much lighter model. Notice the stability between the train and validation set curves compared to the overfitting scenario in our initial model.

A validation accuracy of 64% is performing similarly to other examples of custom CNN implementation for the FER-2013 dataset, which is considered complicated because of mislabeled and noisy data, as well as the low-resolution nature of the black-and-white dataset.

Metrics	Training set	Validation set
Accuracy	75.2%	**64.3%**
Loss	0.68	1.01

Table 2. Final performance metrics

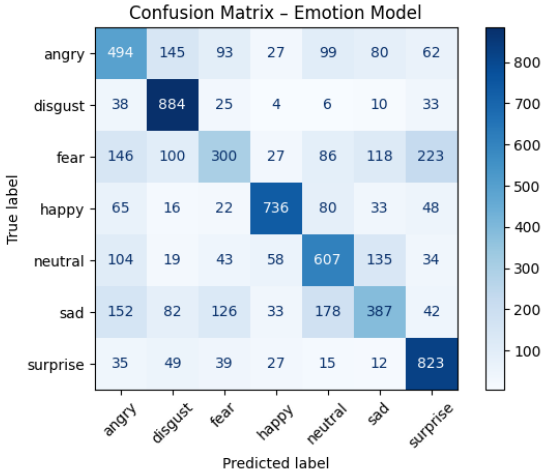


Figure 5. Confusion matrix showing the performance of the final CNN model on the validation dataset. Strong diagonal values indicate accurate predictions for most emotion classes, while off-diagonal entries reveal typical confusions, particularly between *fear*, *surprise*, and *sad*.

6.2 Qualitative results

To better understand how the model performs on real data we visually inspected a couple handfulls of predictions on individual images from the dataset, and compared the predictions to the actual labels.

Where the model performs well Based on this, it seems that the model performs well on emotions - especially exaggerated emotion - with very distinct characteristics and landmarks (e.g. squinted eyes and smile showing teeth in 'Happy'). As such the model seemed to perform well on for example 'Happy' images showcasing a broad smile (as seen in Figure 6. This is corroborated by the 'Happy' class in the confusion matrix (i.e. Figure 5), where it can be observed that 73.6% of the 'Happy' class are correctly predicted by the model.

Figure 6. Five random images with true label 'Happy' and their predicted labels pulled from the dataset to visually inspect single images for where the model seems to have trouble and where it correctly predicts the label.

Where the model falls short The model seems to struggle when the subject does not wear an exaggerated expression on their face, or where the facial landmark of the expression may be more subtle. As such, the model can have trouble in distinguishing between 'Neutral' and 'Sad' with 'Sad' being misclassified as 'Neutral' if the expression is worn more subtly (as can be observed for image 4 and 5 in Figure 7). The predictions are also often misclassifying the image if the expression is of an exaggerated character, but the facial landmarks of the expression are similar to other emotions (as seen in Figure 7 with image 2 and 3, where the model has a hard time distinguishing 'Sad' from 'Fear' (i.e. squinted or closed downward turned eyes, open downward turned mouth, etc.)). This can also be observed in the confusion matrix illustrated in Figure 5, where the most commonly misclassified predictions for 'Sad' are the 'Angry', 'Fear' and 'Neutral' labels.

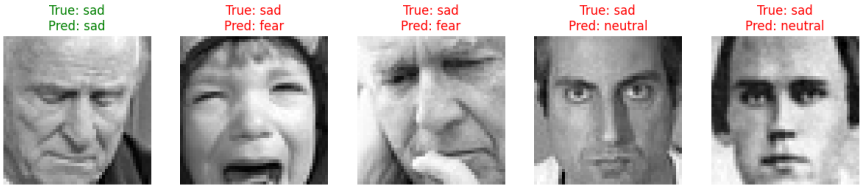


Figure 7. As with 6, 5 random images with true label 'Sad' have been pulled for manual visual inspection by us.

7 CONCLUSION

This project implemented a full deep learning pipeline for facial emotion recognition using the FER-2013 dataset. Through preprocessing, class balancing, augmentation, and a custom CNN architecture, the model was able to learn meaningful emotion related features. The training reached around 75-78% accuracy, with validation accuracy stabilizing near 63-65%, which is consistent with the expected difficulty of the dataset. Some overfitting was observed, reflecting the low image resolution and the visual similarity between certain emotions.

Future improvements could include transfer learning, attention-based architectures, dropping of small less represented classes or more advanced augmentation techniques to improve generalization.

ACKNOWLEDGMENTS

Additional information can be given in the template, such as to not include funder information in the acknowledgments section.

REFERENCES

Kromydas, B. (2023). Convolutional neural network (cnn): A complete guide. *LearnOpenCV*.
SAMBARE, M. (2020). Fer2013. *Kaggle*.
Singh, M. (2020). Image data augmentation for facial recognition. *Medium*.