Assignment - Pattern-Recognition-LRnNN

Part 1 - Logistic Regression Classifier

For this part, I trained a simple **logistic regression** model to tell apart *happy* and *sad* faces from 48×48 grayscale images. Each image was flattened into a 2304-pixel vector and normalized so all pixel values were between 0 and 1.

I started by randomly initializing the weights and bias, then trained the model using **gradient descent** with a small learning rate of 1 × 10^{-6}. Out of the 12 066 total training images, I kept 1 000 aside for validation. The model learned by repeatedly predicting probabilities ($\sigma(XW + b)$), checking how far off they were, and adjusting the weights to reduce the error. I ran the training for **800 epochs**, tracking both the training and validation losses at each step.

Both losses steadily went down, which shows the model was actually learning and not overfitting. The best validation error I got was **0.321** (around **68 % accuracy**), and the final test accuracy on unseen data was **69.3 %**. The learning curves looked smooth and stable, though the training was slow because of the small step size.

Bonus - L2 Regularization

For the bonus part, I added **L2 regularization** with a small $\lambda = 0.01$.

This basically penalizes really large weight values, which helps the model generalize better. After retraining, the best validation error dropped to **0.286** (~ 71 % accuracy), and the test accuracy was **70.3 %**. The loss curves were smoother and closer together, so the regularized version clearly behaved a bit better.

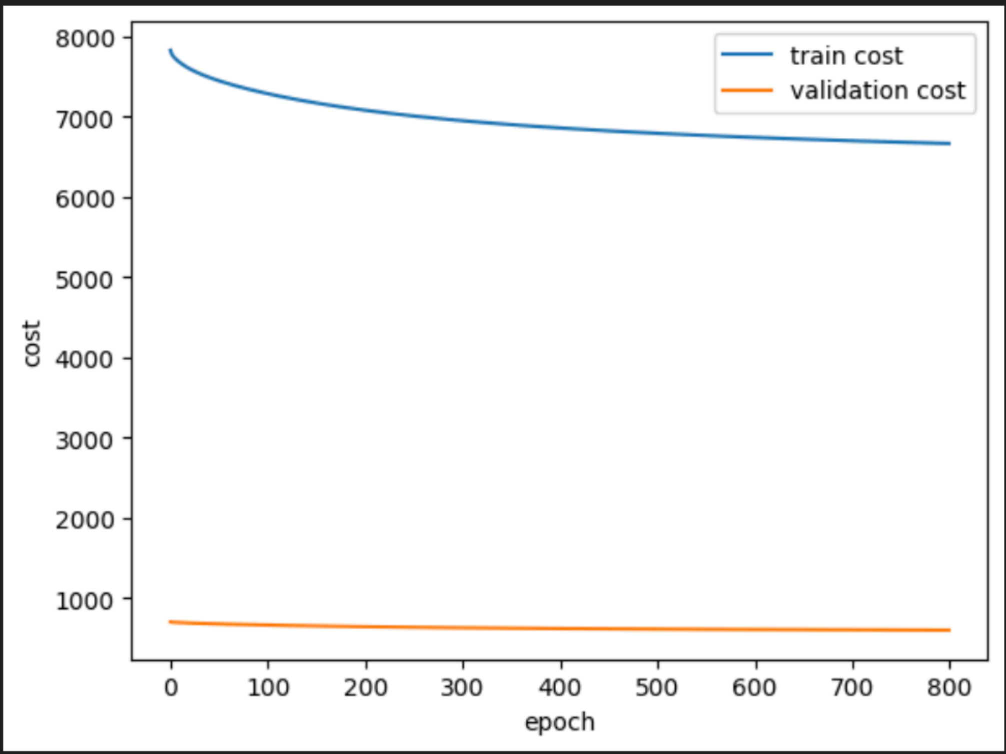
Discussion

Logistic regression is a pretty simple, linear model, so it can’t capture all the complex details in facial expressions - but it still learned something meaningful.

For such noisy data, getting around **70 % accuracy** isn’t bad at all.

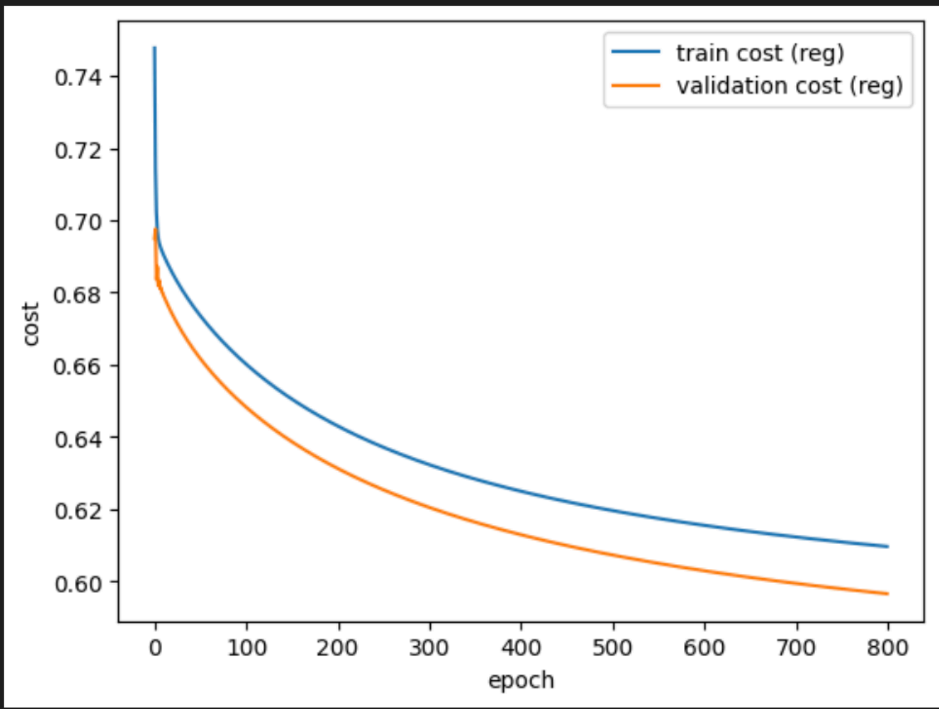
This part was a good warm-up for the more advanced models (SVM and CNN) that come next.

```
epoch 0: train_cost=7822.3605, valid_cost=704.4656, valid_error=0.5350
epoch 100: train_cost=7282.9975, valid_cost=665.1976, valid_error=0.3920
epoch 200: train_cost=7076.7706, valid_cost=644.9415, valid_error=0.3610
epoch 300: train_cost=6946.4435, valid_cost=631.7561, valid_error=0.3540
epoch 400: train_cost=6856.1427, valid_cost=622.4222, valid_error=0.3370
epoch 500: train_cost=6789.5202, valid_cost=615.4233, valid_error=0.3270
epoch 600: train_cost=6738.0228, valid_cost=609.9571, valid_error=0.3200
epoch 700: train_cost=6696.7545, valid_cost=605.5559, valid_error=0.3150
best_validation_error: 0.313
```




```
Accuracy of test set is: 0.6946666666666667
```

```
[REG] epoch 0: train_cost=0.7477, valid_cost=0.6950, valid_error=0.4690
[REG] epoch 100: train_cost=0.6601, valid_cost=0.6482, valid_error=0.3300
[REG] epoch 200: train_cost=0.6430, valid_cost=0.6312, valid_error=0.3250
[REG] epoch 300: train_cost=0.6323, valid_cost=0.6204, valid_error=0.3110
[REG] epoch 400: train_cost=0.6249, valid_cost=0.6129, valid_error=0.3070
[REG] epoch 500: train_cost=0.6196, valid_cost=0.6073, valid_error=0.2980
[REG] epoch 600: train_cost=0.6155, valid_cost=0.6029, valid_error=0.2940
[REG] epoch 700: train_cost=0.6123, valid_cost=0.5994, valid_error=0.2870
[REG] best_validation_error: 0.286
```




```
Test accuracy with L2 regularization: 0.7036666666666667
```

Part 2 – SVM Classification

For this part, I trained **Support Vector Machines (SVMs)** with three different kernels - linear, radial basis function (RBF), and polynomial - to classify the same happy vs. sad faces used in Part 1.

The training data was split 85 / 15 % into a training and validation set, and the final accuracy was reported on the separate test set.

Results:

- The **linear kernel** reached ~ 66 % test accuracy, which is about the same as logistic regression.

- The **RBF kernel** performed best with ~ 74 % test accuracy, showing that a nonlinear boundary fits this dataset better.
- The **polynomial kernel** (degree 3) achieved ~ 73 % test accuracy -also good, but slightly lower than RBF.

  All three confusion matrices are shown below, and the RBF version clearly reduces misclassifications for both classes.

Discussion:

The linear kernel behaves almost identically to logistic regression because both find a single straight decision boundary.

The RBF kernel allows a curved boundary, so it adapts better to complex facial-expression patterns (like differences in mouth curvature or eye region).

The polynomial kernel captures some of that nonlinearity but is more sensitive to tuning the polynomial degree.

Overall, RBF gave the best balance between accuracy and generalization.

(Include the three confusion-matrix heatmaps here.)

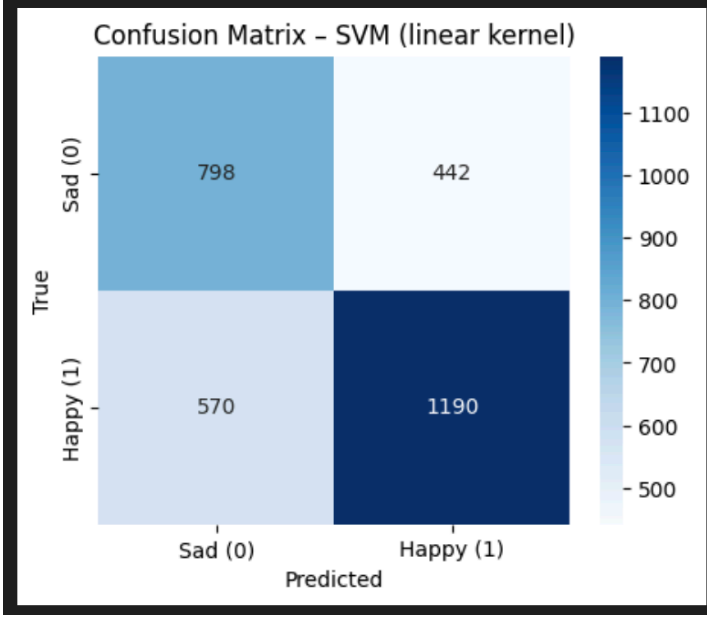
```
------------------------------------------------------------
SVM with linear kernel
------------------------------------------------------------
validation accuracy: 0.707
test accuracy:       0.663

confusion matrix (test):
[[ 798  442]
 [ 570 1190]]

classification report (test):
              precision    recall  f1-score   support

           0       0.58      0.64      0.61      1240
           1       0.73      0.68      0.70      1760

    accuracy                           0.66      3000
   macro avg       0.66      0.66      0.66      3000
weighted avg       0.67      0.66      0.66      3000
```



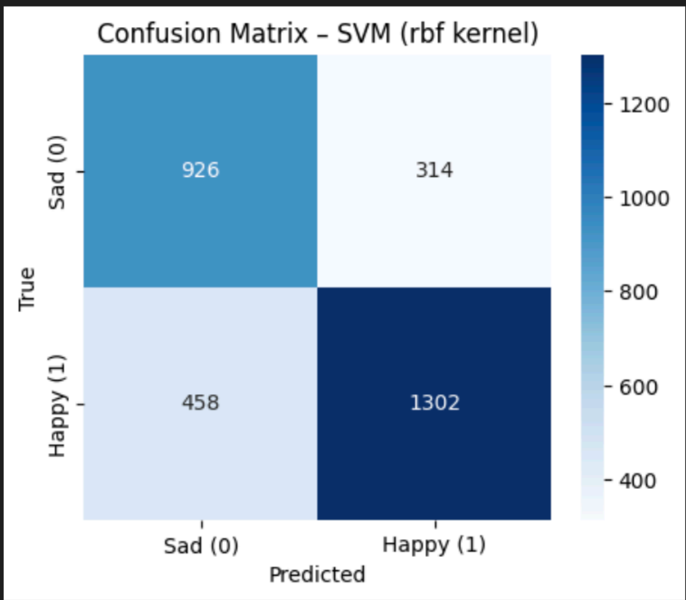
```
------------------------------------------------------------
SVM with rbf kernel
------------------------------------------------------------
validation accuracy: 0.762
test accuracy:       0.743

confusion matrix (test):
[[ 926  314]
 [ 458 1302]]

classification report (test):
              precision    recall  f1-score   support

           0       0.67      0.75      0.71      1240
           1       0.81      0.74      0.77      1760

    accuracy                           0.74      3000
   macro avg       0.74      0.74      0.74      3000
weighted avg       0.75      0.74      0.74      3000
```



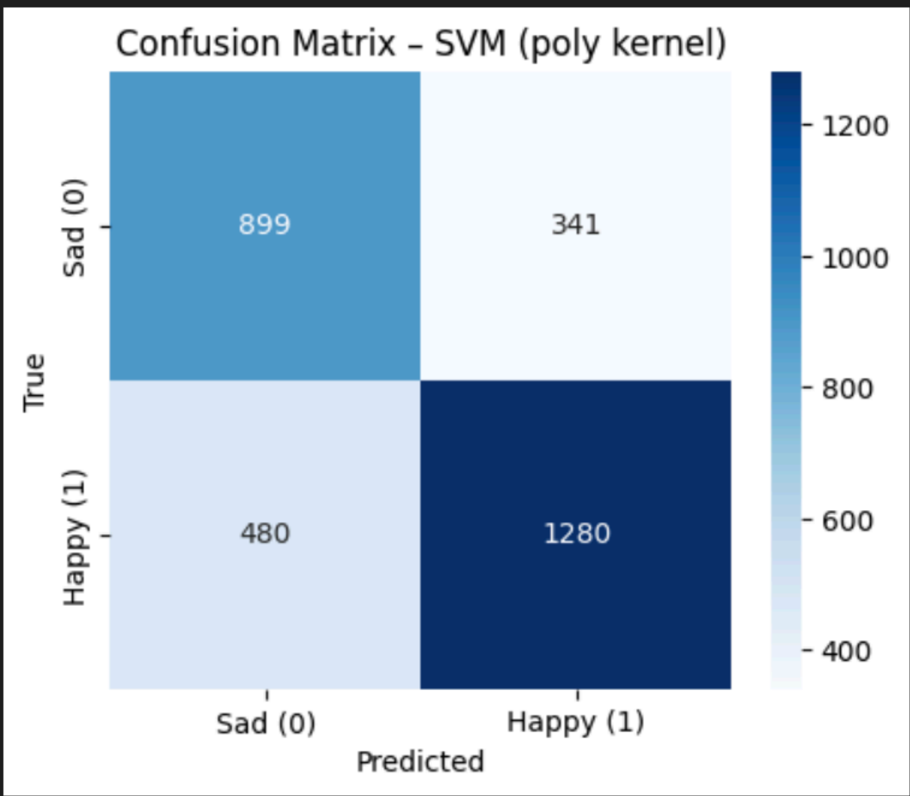
```
------------------------------------------------------------
SVM with poly kernel
------------------------------------------------------------
validation accuracy: 0.786
test accuracy:       0.726

confusion matrix (test):
[[ 899  341]
 [ 480 1280]]

classification report (test):
              precision    recall  f1-score   support

           0       0.65      0.72      0.69      1240
           1       0.79      0.73      0.76      1760

    accuracy                           0.73      3000
   macro avg       0.72      0.73      0.72      3000
weighted avg       0.73      0.73      0.73      3000
```



Part 3 – Freeform CNN Classification

For the final part, I implemented a **Convolutional Neural Network (CNN)** to classify the same *happy* and *sad* face images.

Each 48 × 48 grayscale image was reshaped to a 48×48×1 tensor and normalized to values between 0 and 1. The CNN architecture contained three convolutional layers (with 32, 64, and 128 filters) followed by two fully connected layers.

The model used the **ReLU** activation function, **max-pooling** after the first two convolutional layers, and was trained with the **Adam** optimizer (learning rate = 1×10^{-3}) and **binary cross-entropy loss** for 12 epochs, using a batch size of 64.

Training was done on my MacBook Pro's M-series GPU through Apple's Metal backend (MPS).

The model converged quickly: training accuracy reached 100 %, while validation accuracy stabilized around **89 %**, indicating good generalization without heavy overfitting.

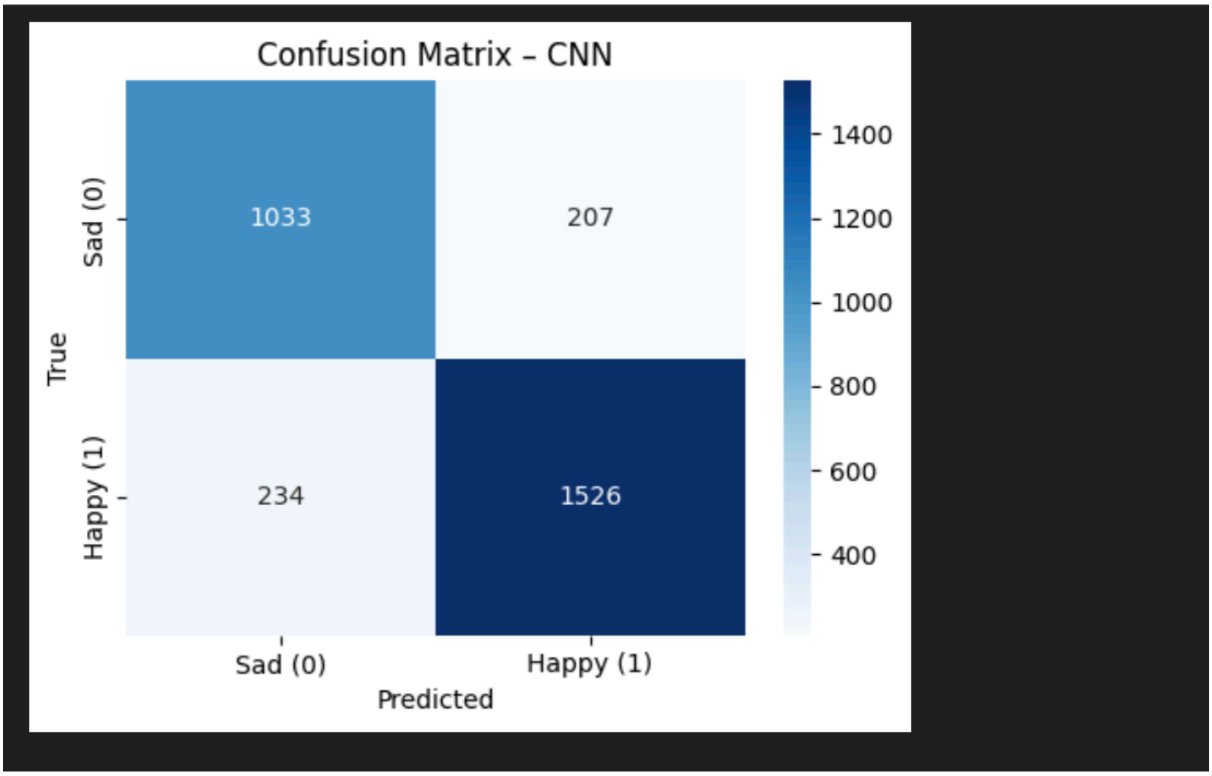
On the unseen test set, the CNN achieved a **test accuracy of 85.3 %**.

The confusion matrix shows that most images were classified correctly (1033 / 1240 sad and 1526 / 1760 happy), and both precision and recall were around 0.85.

Compared to the simpler models from Parts 1 and 2 - logistic regression (~70 % accuracy) and SVM with RBF kernel (~74 %) - the CNN performed substantially better.

This improvement makes sense because convolutional layers learn **spatial features** such as edges, eyes, and mouth shapes that linear models cannot capture.

Overall, the CNN provides the best trade-off between training accuracy and real-world generalization for this dataset.


Confusion Matrix – CNN
1033
207
234
1526
1400
1200
1000
800
600
400
Sad (0)
Happy (1)
True
Sad (0)
Happy (1)
Predicted