ASSIGNMENT II – Scott McHaffie

Convolutional Neural Networks

SH2150 Machine Learning in Physics

November 2025

In this assignment, you will investigate the progression from hand-crafted image features to deep learned representations using convolutional neural networks (CNNs). You will work with one of the provided datasets (galaxies, brain-stroke, or histopathology), each containing three classes. Each dataset `.zip` file contains more information about the respective data.

Your tasks are to

- extract hand-crafted features from images,
- perform unsupervised clustering and analyze the results,
- train a feed-forward neural network on the extracted features,
- train a convolutional neural network classifier using the full images,
- compare the performance of the models using confusion matrices and t-SNE plots.

Hint: Normalization of Features Before training your models, it is important to normalize the features. Different features may have very different scales (for example, mean pixel intensity may range from 0–255, while variance or edge density may lie in a much smaller range). Models such as K-means and MLPs are sensitive to feature scaling, and unnormalized data can lead to poor clustering and slow or unstable training. Normalization ensures that all features contribute comparably during learning. A common approach is to apply standardization (subtract the mean and divide by the standard deviation for each feature), computed across the training set and then applied to both training, validation and test data. Another approach is min-max normalization, where you map all features to the range $[0, 1]$ or $[-1, 1]$.

1. Hand-crafted Feature Extraction and Unsupervised Analysis

(a) (2p) Extract 3–5 meaningful but simple hand-crafted features (e.g. intensity mean, standard deviation, edge density, etc.) from each image in your dataset. Briefly describe your choices and expected usefulness for classifying the images into the three categories.

Answer: I chose to work with the **brain-stroke** image dataset and I extracted 5 hand-crafted features from the dataset. The 5 features are listed below, and I give a short explanation of my reasoning for each.

1. Mean pixel intensity: The mean pixel intensity of the images is a relevant parameter because, from what I saw in the dataset images, the bleeding typically shows up as a lighter white color, while the lack of blood (ischemia) appears as a darker black. This is consistent with how this behavior is normally observed in CT scans.
2. Standard deviation: This captures how much pixel intensities vary within the image. I expect both the bleeding and ischemic brains to have a larger standard deviation than the normal brain images.
3. Mean gradient: The mean gradient is a good indicator of how how quickly intensities change from pixel to pixel, making it useful to detect abnormalities in the brain images (like bleeding or ischemia) where pixel values can change quite quickly.
4. Entropy: I calcualted the entropy of the images using the `shannon_entropy` function from `skimage.measure`. A high entropy represents an image with many different pixel values, while a low entropy is characteristic of an image with a limited number of pixel values, where one pixel value dominates.
5. Skewness: This is a good measure of how asymmetric the images might be, making it useful for detecting bleeding or ischemia (as long as the occurences are not directly in the center of the brain image).

(b) (2p) Perform K-means clustering (either define it yourself or use the `KMeans` algorithm from `sklearn.cluster`) with $K = 3$ clusters on the training data. Use majority-voting on the labels of the fitted data points in each cluster to assign a label to it. See `https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html`.

(c) (2p) For each point in the test set, predict what cluster/category it belongs to and visualize the data using t-SNE in $\mathbb{R}^2$. Discuss how well clusters align with true labels. Also, report the confusion matrix.

Answer: I predicted which category each point in the test set corresponds to and visualized the data using t-SNE in $\mathbb{R}^2$, with the results shown in Figure 1. In this figure we can see the K-means clustering with three distinct clusters.

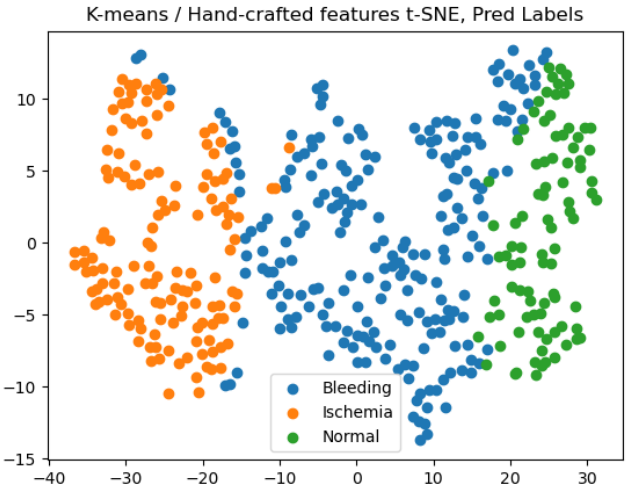


Figure 1: The K-means clustering of the image data is visualized using t-SNE in $\mathbb{R}^2$.

I then compared the predicted categorization to the true labels by visualizing the true labels using t-SNE in $\mathbb{R}^2$. The true labels are shown in Figure 2, and it is clear that the K-means clusters do not align very well with the true labels. The effectiveness of K-means clustering on this dataset is limited, given the complicated and largely unclustered nature of the dataset.

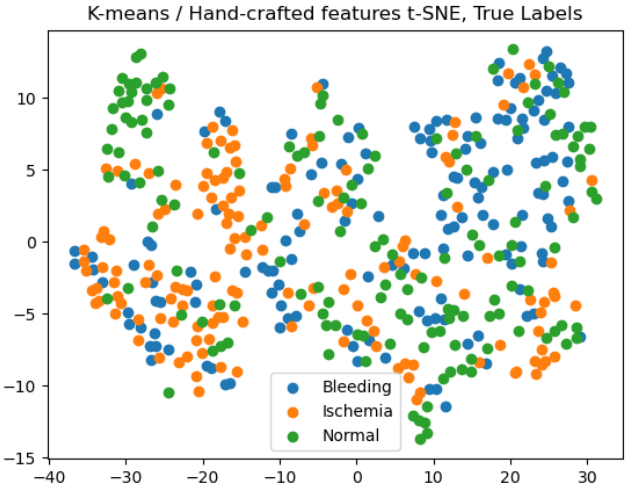


Figure 2: The true labels of the image data is visualized using t-SNE in $\mathbb{R}^2$.

Finally, I computed the confusion matrix of the K-means clustering method, shown in Figure 3. The confusion matrix captures the largely unsuccessful prediction of the K-means fit, with a calculated accuracy score of 0.42.

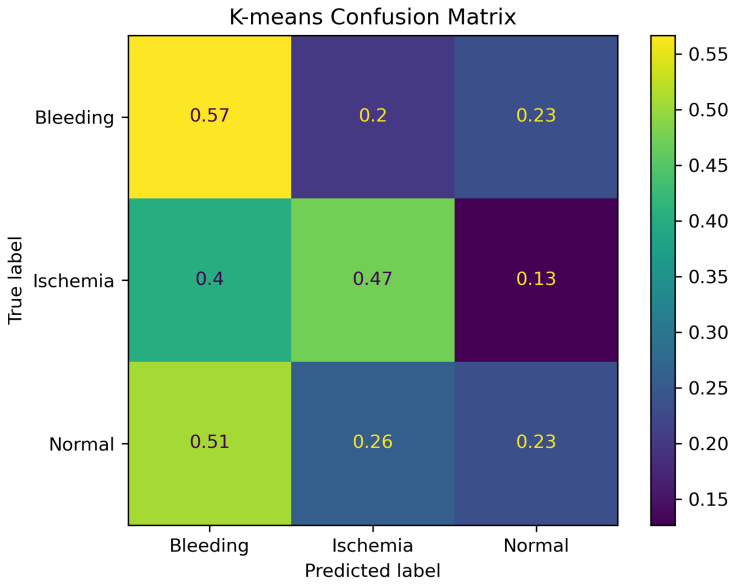


Figure 3: The confusion matrix for the K-means clustering.

2. MLP Classifier on Extracted Features

(a) (2p) Define an MLP classifier similar to the network in the previous assignment sheet (Assignment I). Make sure the output dimension is three and switch to `torch.nn.Softmax()` activation in the final layer. Use one-hot encoded targets. You might want to play around a bit with the hidden activation functions (not the final one).

Answer: I defined an MLP classifier according to the instructions above, and used the `torch.nn.ReLU()` as the hidden layer activation functions. I played around with a few other activation functions (such as `torch.nn.Tanh()`) but I did not see any large improvements in classification, so I stuck with the activation function `torch.nn.ReLU()`.

(b) (2p) Train the MLP using cross-entropy and the Adam optimizer, on the extracted features in 1(a) from the train set. Validate the model performance on the validation set during training; plot a loss curve for the train set and val set respectively, as a function of epoch. Does the model generalize well?

Answer: I then trained the MLP using my cross-entropy loss function from Assignment 1, and the Adam optimizer. I trained on the extracted feature training dataset from 1(a) and validated with the validation set for 500 epochs. The loss curve for both the training set and validation set are shown across the epochs in Figure 4.

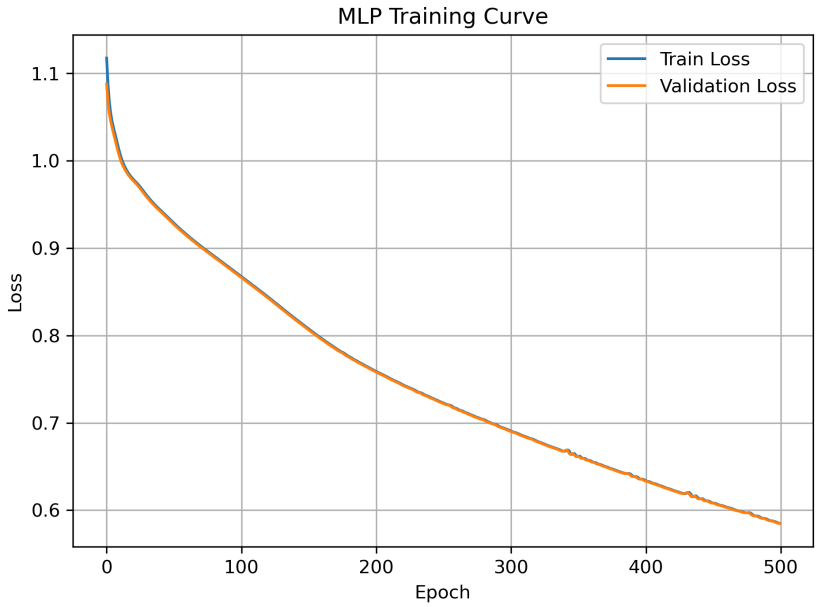


Figure 4: The training (blue) and validation (orange) loss curves as a function of epoch for the MLP classifier.

The model generalizes quite well and the loss curves for the training and validations sets are nearly overlapping across all epochs.

(c) (2p) Evaluate the model on the test set and report a confusion matrix. For each data point, extract the output from the final layer of the network, project it to $\mathbb{R}^2$ using t-SNE and report a plot. Which classes were hardest to distinguish and why?

Answer: I evaluated the model on the test set and calculated the confusion matrix, shown in Figure 5. The MLP classifier performed much better than the K-means clustering, with an accuracy score of 0.62.

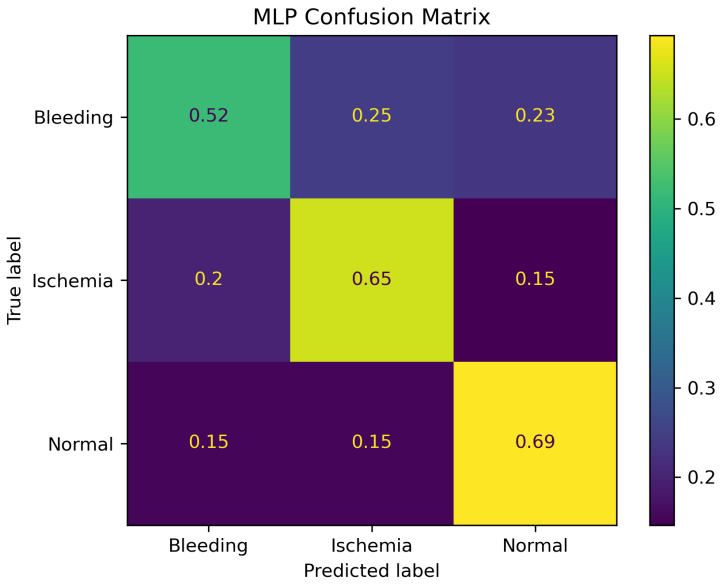
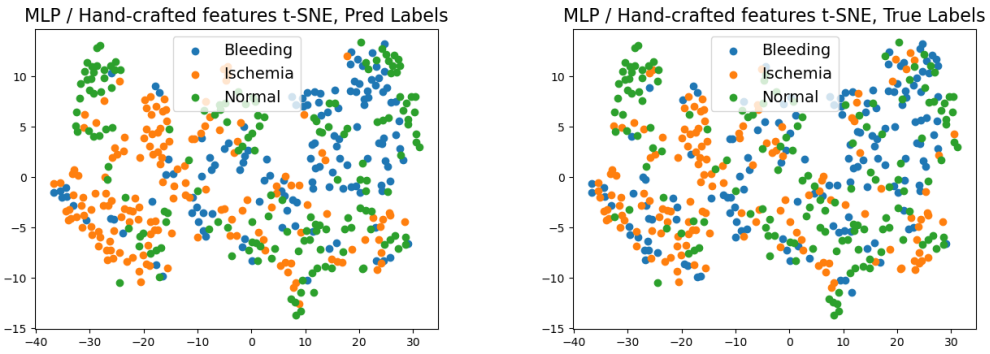


Figure 5: The confusion matrix for the MLP classifier.

Additionally, I projected the output from the final layer of the network to $\mathbb{R}^2$ and visualized the data using t-SNE. These results are shown in Figure 6a and can be compared to the true labels in Figure 6b.



(a) Predicted labels.

(b) True labels.

Figure 6: The image labels predicted by the MLP classifier is visualized using t-SNE in $\mathbb{R}^2$ and compared to the true image labels.

From Figure 5 it is clear that the hardest class for the MLP classifier to distinguish was the Bleeding class. This could be because the hand-crafted features I selected favored the classification of the Ischemia and Normal classes or due to the fact that bleeding appears as light colored regions in the images, which is similar to other normal brain structures (such as veins).

3. CNN Classifier on Full Images

(a) (1p) Compare the number of trainable parameters in the following two layers. Assume the image is single–channel (grayscale).

- A fully connected (linear) layer that takes the flattened 128×128 image as input and outputs 512 neurons.
- A convolutional layer with 64 filters, each of spatial size 3×3 (stride 1, padding such that the spatial dimensions are preserved).

For each case, write the formula you use and compute the total number of parameters. Finally, explain what your result implies about the risk of overfitting when using fully-connected layers versus convolutional layers on images.

Answer: Let's start with the fully connected layer that takes an input vector of size $128 \times 128 = 16\,384$. This fully connected layer has 512 neurons and each neuron has 16 384 inputs, meaning there are 16 384 trainable weights for each neuron. Aditionally, each neuron has one trainable bias parameter, yielding a total number of trainable parameters

$$\begin{aligned} \text{num params} &= \text{num weights} \times \text{num neurons} + \text{num biases} \\ &= 16\,384 \times 512 + 512 \\ &= 8\,389\,120. \end{aligned}$$

With the convolutional layer that has 64 filters, each of spatial size 3×3 and a stride of 1, each filter has a total of $3 \times 3 = 9$ weights and one bias term, yielding a total number of trainable parameters

$$\begin{aligned} \text{num params} &= \text{num filters} \times \text{num weights per filter} + \text{num biases} \\ &= 64 \times 9 + 64 \\ &= 640. \end{aligned}$$

Now we can cleary see that there are many more trainable parameters for the fully-connected layers than for the convolutional layers which makes the fully-connected layers more susceptible to overfitting, especially as the fully-connected layers do not capture the relationships between pixels like the convolutional layers do.

(b) (2p) Implement and train a CNN classifier. The CNN should input the full images (128×128 pixels) and output one probability for each class, similar to the MLP in exercise 2 above. The architecture can be defined according to Figure 7, but you are free (not mandatory) to design your own architecture if you describe your design choices. Each convolutional layer should be followed by a non-linear activation function like ReLU, and a 2×2 max-pooling. In the final stage of the network, the features are flattened and two fully connected (linear) layers are used to produce the class probabilities. For training, use

cross-entropy loss, Adam optimizer, and validate the model performance on the validation set in each epoch. Provide loss curves as a function of epoch, as in 2(b).

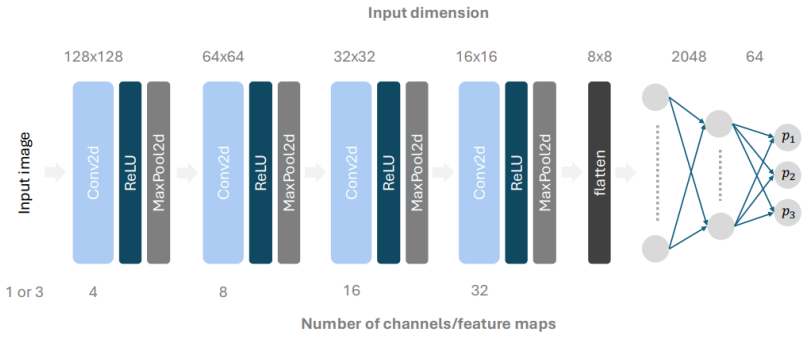


Figure 7: CNN architecture example. The network consists of four convolutional blocks consisting of a `Conv2d` layer with same padding, ReLU activation, followed by a `MaxPool2d` layer respectively. The final (MLP) part of the network maps the flattened output from the CNN blocks through a first fully connected (linear) layer followed by a ReLU activation. The final fully connected layer is followed by a softmax activation to produce the final class probabilities p_1, p_2, p_3.

Answer: I implemented and trained a CNN classifier according to Figure 7 and the instructions above. I input the full images and trained the CNN on the training dataset and validated with the validation set. I computed the validation and training loss across 6 epochs and plotted the loss curves in Figure 8. I picked the value of 6 epochs by first running the training loop for 10 epochs and selecting the largest epoch number for which the validation loss curve did not begin to flatten.

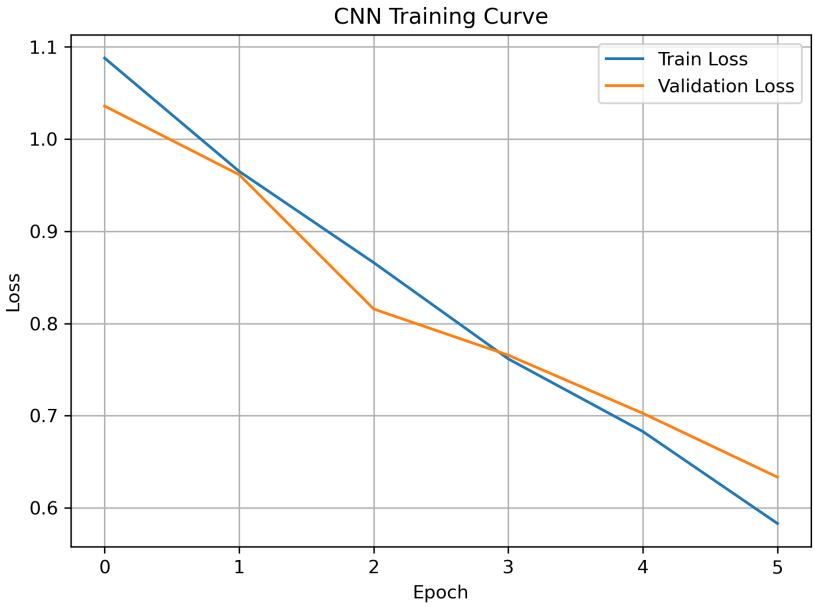


Figure 8: The training (blue) and validation (orange) loss curves as a function of epoch for the CNN classifier.

(c) (1p) Evaluate the performance of your CNN classifier by first computing the confusion matrix on the test set and briefly commenting on which classes are most often misclassified.

Answer: I then evaluated the performance of my CNN classifier by computing the confusion matrix on the test set. The confusion matrix is shown in Figure 9 and the CNN classifier was better than both the MLP classifier and the K-means clustering with an accuracy score of 0.74. The CNN classifier was biased towards predicting the class as Normal, giving a very high fraction of TP for the Normal class, but also often misidentifying Ischemia and Bleeding classes as Normal too.

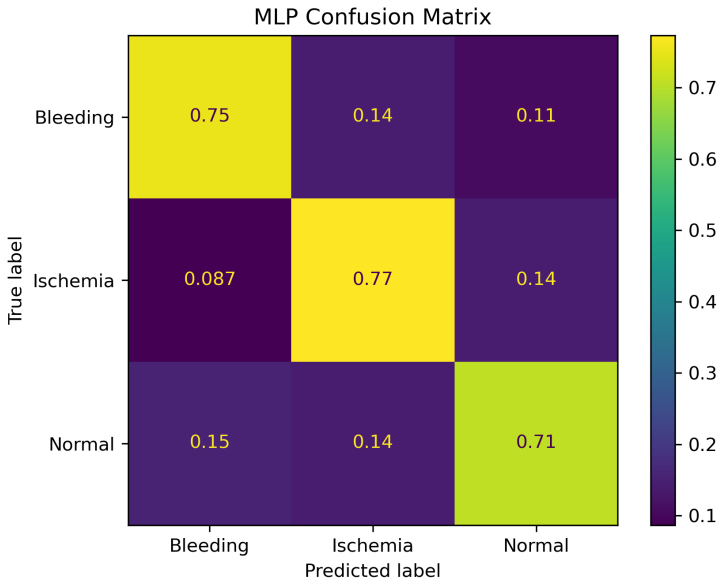
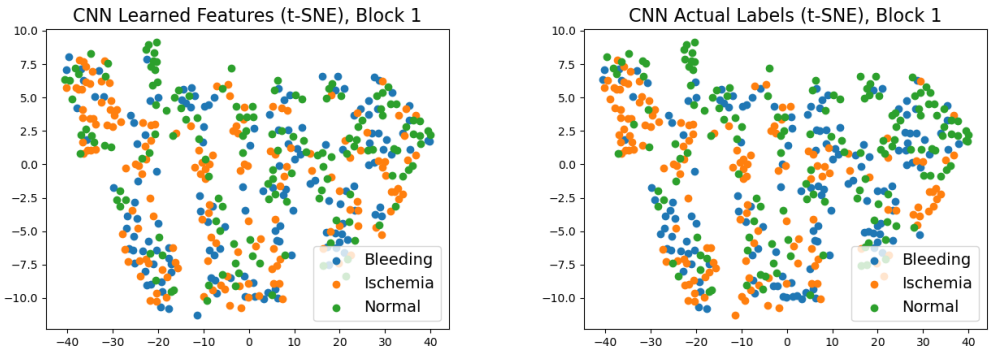


Figure 9: The confusion matrix for the CNN classifier.

(d) (2p) For each convolutional block in the network, extract the feature representations of all samples in the validation set and use t-SNE to project these features into $\mathbb{R}^2$. Create one two-dimensional visualization per block. Describe how the structure of these feature spaces evolves through the network, with particular attention to how the intra-class compactness and inter-class separability change as depth increases. See `https://scikit-learn.org/stable/modules/generated/sklearn.manifold.TSNE.html`.

Answer: For each convolutional block in the network, I extracted the feature representations of all samples in the validation set and used t-SNE to project these into $\mathbb{R}^2$.

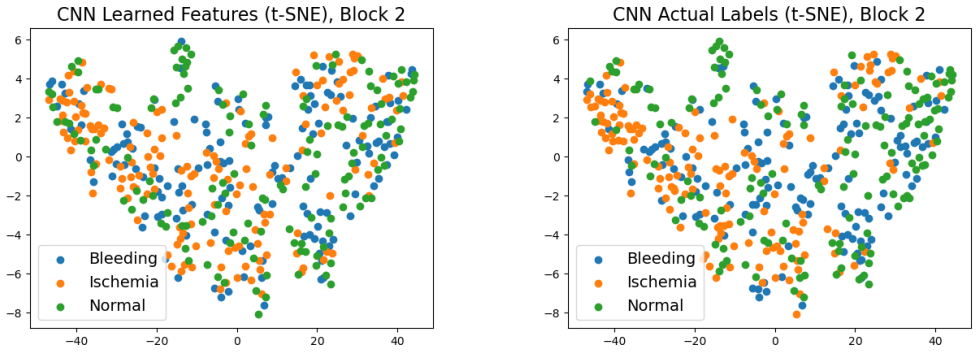
The t-SNE plot for the predicted classification of the data after convolutional block 1 is shown in Figure 10a, next to the t-SNE plot for the true data classification in Figure 10b. Here, we see a large overlap between data points (low inter-class separability) and not many well defined or tight clusters (low intra-class compactness).



(a) Predicted labels. (b) True labels.

Figure 10: The image labels predicted by the CNN classifier after the first convolutional block is visualized using t-SNE in $\mathbb{R}^2$ and compared to the true image labels.

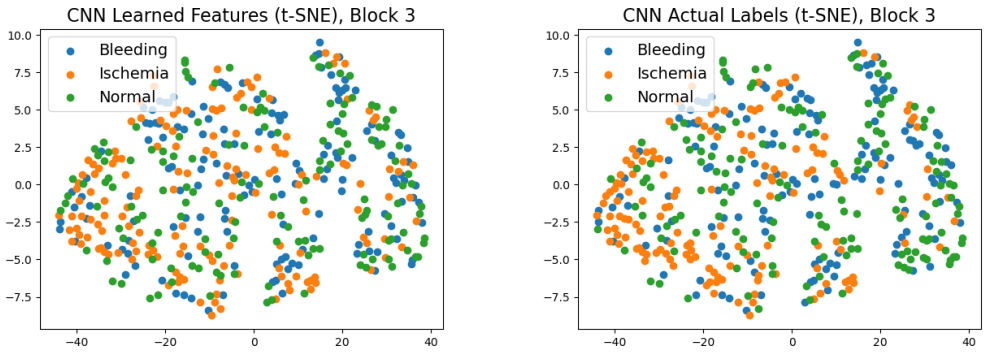
The t-SNE plot for the predicted classification of the data after convolutional block 2 is shown in Figure 11a, next to the t-SNE plot for the true data classification in Figure 11b. Here we still have mostly overlapping data points and no well defined and separated clusters.



(a) Predicted labels. (b) True labels.

Figure 11: The image labels predicted by the CNN classifier after the second convolutional block is visualized using t-SNE in $\mathbb{R}^2$ and compared to the true image labels.

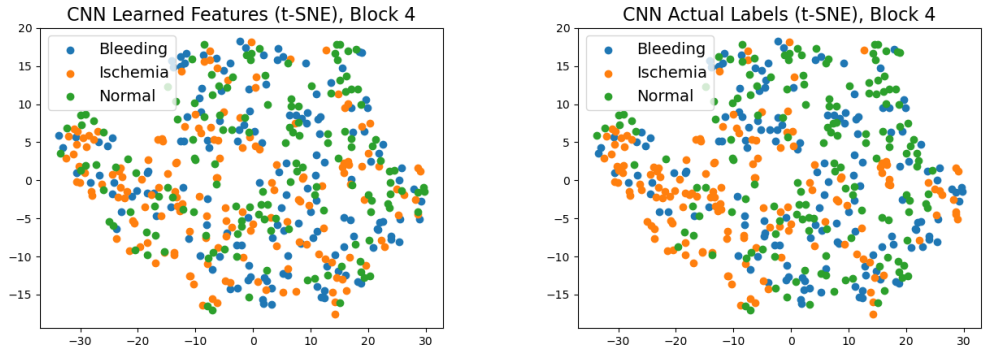
The t-SNE plot for the predicted classification of the data after convolutional block 3 is shown in Figure 12a, next to the t-SNE plot for the true data classification in Figure 12b. Here there is still overlap between the data points and the three different classes, but we do also start to see some small clusters forming.



(a) Predicted labels. (b) True labels.

Figure 12: The image labels predicted by the CNN classifier after the third convolutional block is visualized using t-SNE in $\mathbb{R}^2$ and compared to the true image labels.

Finally, the t-SNE plot for the predicted classification of the data after convolutional block 4 is shown in Figure 13a, next to the t-SNE plot for the true data classification in Figure 13b. There is a little bit of clustering but we still see overlap between the different classes and it appears that the inter-class separability and intra-class compactness are not large, likely what results in the classification accuracy of 0.74.



(a) Predicted labels. (b) True labels.

Figure 13: The image labels predicted by the CNN classifier after the fourth (last) convolutional block is visualized using t-SNE in $\mathbb{R}^2$ and compared to the true image labels.

(e) (2p) Finally, compare these observations to the separability observed in the hand-crafted feature space used for K-means, as well as to the learned feature space of the MLP classifier. Discuss why the CNN may achieve stronger class separation and more robust classification performance.

Answer: The greatest separibility was achieved in the K-means t-SNE visualisation, however this clustering was far from perfect and resulted in a low accuracy of classification, of only 0.43. With both MLP classifier and the CNN

classifier, we observed low class separation. The CNN classifier performed the best, yielding a classification accuracy of 0.74.

CNNs are expected to observe stronger class separation due to their ability to learn and extract features from images.

Final question: Did you use an AI tool (other than the machine learning models you trained in this exercise) for anything else than information searching, when solving this problem set? If so, please write a brief statement of how you used AI.
Answer: Here is my AI usage for each task.

- Task 0: I used Chat-GPT to generate the latex code for the provided assignment PDF. I then edited this latex code and filled in my own answers beneath each task.
- Task 1: I used Chat-GPT to help me determine which hand-crafted features I should extract and used it to help me extract the mean gradient and entropy features.
- Task 2: I used Chat-GPT to help me determine some potential reasons as to why the Bleeding class was the hardest for the MLP classifier to classify.
- Task 3: I used Chat-GPT to help me interpret my results, and understand why I wasn't seeing as much class separation as I expected.

Total number of points: 20

Remember to motivate your answers wherever applicable.

Good luck!