# Applied Machine Learning - Assignment 5

**Ananthan Srinath Adhvait**
**Chalmers University of Technology**
adhvait@chalmers.se

**Lee Yu Xuan**
**Chalmers University of Technology**
yuxua@chalmers.se

**Teo Xuan Ming**
**Chalmers University of Technology**
xuanm@chalmers.se

**Abstract**

Skin cancer, particularly melanoma, is a major public health concern due to its high incidence and potential for metastasis if not detected early. In this assignment, we investigate the efficacy of various machine learning techniques for automatically classifying skin lesions, with a focus on distinguishing between melanoma and nevi. We utilize a subset of the 2018 ISIC challenge dataset and explore techniques such as normalization, residual connections, data augmentation, and transfer learning to enhance classification accuracy. A Convolutional Neural Network (CNN) architecture is implemented and trained on the training set, with evaluations conducted on separate validation sets.

## 1 Introduction

Skin cancer, including melanoma, represents a significant global health burden with rising incidence rates and substantial morbidity and mortality. For example, incidence in Europe is about 25 cases per 100,000 population, while in Australia it reaches a rate of 60 new cases per 100,000 (Conforti & Zalaudek, 2021). Melanoma has a high potential for metastasis if not detected and treated early, underscoring the critical need for accurate and timely diagnosis. Visual inspection of skin lesions is a primary method for identifying suspicious growths, with melanoma typically exhibiting distinct characteristics such as irregular shape, color variation, and larger size compared to benign nevi.

Advancements in machine learning techniques offer promising avenues for improving the accuracy and efficiency of skin cancer diagnosis through automated image classification. The International Skin Imaging Collaboration (ISIC) challenges have been pivotal in driving research and innovation in this field. They provide curated datasets and benchmarks for evaluating the ML algorithm's performance in classifying skin lesions.

In this study, we leverage a processed subset of the 2018 ISIC challenge dataset to explore and compare various ML solutions for automatically classifying skin lesions into melanoma and benign nevi categories. We assessed the efficacy of different techniques, including batch normalization (BN) methods, data augmentation through random grayscalling and flipping and transfer learning. We also implement a Convolutional Neural Network (CNN) architecture with Pytorch's "nn.module" which gives us a greater degree of freedom and ease of debugging.

Through this investigation, we aim to contribute insights into the optimal approaches for automated skin cancer diagnosis, ultimately facilitating early detection, timely intervention, and improved patient outcomes. By utilizing ML and image analysis, we aim to enhance the efficiency and accuracy of skin cancer diagnosis, thereby addressing a crucial healthcare challenge with significant societal impact.

## 2 Methodology

Our approach consists of two key steps: data preprocessing, model setup, and model training.

### 2.1 Data Preprocessing

```
train_folder = ImageFolder('/content/a5_data/train', transform=torchvision.transforms.ToTensor())
train_loader = DataLoader(train_folder, batch_size=24, shuffle=True)
val_folder = ImageFolder('/content/a5_data/val', transform=torchvision.transforms.ToTensor())
val_loader = DataLoader(val_folder, batch_size=24, shuffle=True)
```

Before training the classifiers, we need to preprocess our images such that the model can understand the images and learn from them. The model is unable to interpret the images as it is, we would need to convert it into a manner that the model can interpret. As such, we used the transform = torchvision.transforms.ToTensor() to convert the images in the folders into Tensors. Additional layers of transformation such as random flipping and grayscaling were defined and overwritten with Pytorch transform's compose API. This API call allows us to easily do data augmentation on the images.

Data augmentation is the process of artificially generating new data from existing data to train a machine-learning model. Such examples of data augmentation for images are rotating the image by a certain angle, shifting the image horizontally or vertically, or resizing the image.

Data augmentation is beneficial for the CNN model when we are training it as it increases the robustness of the model by exposing the model to different variations of the same image, we allow the model to recognize the image from different perspectives and orientations. Another benefit of data augmentation allows the model to have a better generalization of images instead of the model memorizing specific patterns from the training dataset of images. This prevents the model from being overfitted.

The DataLoader() code allows us to create batches of training data for training or validation. In this case, we have specified the batch size to be 240.

### 2.2 Model Setup

The model architecture we have decided on is a 3-layer convolution and max pool, followed by 2 linear layers. Our convolution layers use a kernel size of 3 and are then parsed through batch normalization. Then, ReLU is used as the activation function before the maxpool layer with a kernel size of 2 and stride 2. Finally, 2 linear layers are used.

The model employs the AdamW optimizer, which is instantiated and initialized with the model parameters. The 'model.parameters()' function returns all of the model's parameters (weights and biases) that the optimizer will optimize. AdamW was chosen due to its effectiveness in training neural networks, particularly in scenarios with large datasets and complex architectures.

The model also employs the Cross-Entropy Loss function, which is commonly used in multi-class classification tasks. The 'CrossEntropyLoss()' function initializes the loss criterion that will be used during training. This loss function then calculates the loss between the predicted class probabilities and the actual class labels, guiding the optimization process to reduce this loss and improve the model's classification performance.

The above layers were extended to a VGG16 transfer learning model where the last of the 16 layers was overridden by our customized model architecture.

### 2.3 Model Training

Model training consists of several key steps that optimize the neural network architecture for accurate image classification. First, the dataset is loaded using PyTorch's DataLoader, which allows for efficient batch processing during training. The training loop then iterates through 10 epochs, with each epoch containing mini-batches of 240 images. A forward pass is also performed, in which input images are fed into the model and predictions are generated. These predictions are then compared to the ground truth labels via the Cross-Entropy Loss function. The AdamW optimizer is then used to update the model parameters by backpropagating loss gradients through the network. This process iterates over the entire training dataset, gradually adjusting the model's parameters to reduce loss and increase classification accuracy.

Additionally, during training, the model's performance on the validation dataset is evaluated regularly to monitor its generalization ability and prevent overfitting. Finally, after completing all epochs, the training process is complete, and the trained model is ready for evaluation and deployment on previously unseen data. Hyperparameters such as batch sizes, epochs, learning rate, optimization algorithm, and loss function were tweaked along the way based on the validation accuracy, and loss.

## 3 Results and Discussion

### 3.1 Model Performance

| Model Number | Normalization | Transformation | Loss | Validation Accuracy |
|---|---|---|---|---|
| 1. (CNN) | None | None | 0.5221 | 0.7658 |
| 2. (CNN) | Batch | Random flipping | 0.4239 | 0.7523 |
| 3. (CNN) | Batch | Random flipping Random grayscaling | 0.4775 | 0.7804 |
| 4. (VGG16 Transfer Learning) | Batch | Random flipping Random grayscaling | 0.3581 | 0.8371 |

Model number 3 managed to get a score of 0.7753 on the blind test.

Model number 4 managed to get a score of 0.8279 on the labelled test set released on March 7th.

### 3.3 Discussion

As shown in Table 3.2, the introduction of BN has allowed the model to perform better on the validation set. BN normalizes the activations for each mini-batch and forces them to have a mean of zero and a standard deviation of one which facilitates learning across the layers. BN also improves the gradient flow as it prevents exploding or vanishing gradients which allows gradients to propagate effectively during backpropagation. Lastly, BN reduces overfitting as normalization is technically adding noise to the data.

Likewise, data augmentation improves the model performance as it generates noise and randomness in the data, reducing the likelihood of overfitting.

In Model 4, we see the benefits of transfer learning in Model 4 as there was a huge bump in performance. This is because VGG16 has been trained on a massive dataset and has been generalised to be used for generic computer vision tasks. With transfer learning, the pre-trained layers are used as feature extractors especially when we have limited data like in this assignment. Moreover, the pre-trained features act as a sort of regularizer and help reduce the risk of overfitting. Most importantly, transfer learning helps reduce our training time greatly.

### 3.3 Limitations and Future Directions

One limitation is that the CNN may need help to generalize the lesions significantly different from those in the training dataset. This includes lesions from different populations, unusual shapes, or lesions captured under different imaging conditions.

It is also hard to interpret the decision of CNN when it categorizes the image into cancerous or non-cancerous. Interpretability is important for gaining the trust of clinicians, if they do not understand the decision-making of the CNN, they would not be so likely to trust the model to do the classification.

Given more time and computational resources, we can improve the accuracy of the CNN model by improving our model architecture by increasing the depth of the CNN. This is because CNN generally performs better as the depth increases, up to a certain threshold (openreview, 2021).

We could also experiment with more ways of augmenting the data to generate more training data for the CNN.

Another way we can improve this is to combine CNN with any classification model to improve the model's accuracy and interpretability. Such examples could be a decision tree model that explicitly states the feature and decision why an image is being classified as cancerous or not-cancerous.

## 4 Conclusion

In conclusion, image processing and classification tasks are complex given the popularity of CNN in the past few years. This assignment has allowed us to explore how the CNNs are constructed and used in image classification. However, it also shows us how difficult it was for us to choose the hyperparameters for the models such that we can train the model at a reasonable speed without sacrificing accuracy,

Citations:


1. Conforti, C., & Zalaudek, I. (2021). Dermatology Practical and Conceptual. *Epidemiology and Risk Factors of Melanoma: A Review*. https://www.ncbi.nlm.nih.gov/pmc/articles/PMC8366310/
2. Openreview, (2021). *Do Deeper Convolutional Networks Perform Better*. https://openreview.net/pdf?id=rYt0p0Um9r#:~:text=Increasing%20depth%20beyond%20a%20critical,network%20(shown%20in%20red).