**Hong Kong University of Science and Technology**

**COMP 4211: Machine Learning Spring**

**2025**

**Programming Assignment 2**

Andreas Wold Sløgedal

Student ID: 21196840

Contents

[**Introduction** 3](#_Toc195035948)

[**Part 1: Classification Task** 3](#_Toc195035949)

[[Q1] Samples from Training Dataset 3](#_Toc195035950)

[[Q2] Keras Model Implementation 4](#_Toc195035951)

[[Q3] Usage of Adding Gaussian Noise 4](#_Toc195035952)

[[Q4] Model Training 5](#_Toc195035953)

[[Q5] Model Accuracy 5](#_Toc195035954)

[[Q6] Accuracy of Regularized Model 6](#_Toc195035955)

[**Part 2: Generation Task** 6](#_Toc195035956)

[[Q7] Energy-Based Evaluation of Images 6](#_Toc195035957)

[[Q8] Visualization Buffer Samples 7](#_Toc195035958)

[[Q9] JEM Training and Loss Analysis 8](#_Toc195035959)

[[Q10] JEM Accuracy Analysis 9](#_Toc195035960)

# **Introduction**

This report presents the implementation of a Wide ResNet (WRN-16-4) architecture for image classification. The project involves creating a custom ResNet block, implementing the WRN model, and developing a trained model with Guassian noise augmentation. Furthermore, there will be used regularization techniques to improve model performance. The results will be analyzed through training metrics and accuracy comparisons. Ultimately, the main objective is to create a model which generalizes well to unseen images and categorizes images into their respective categories.

# **Part 1: Classification Task**

## [Q1] Samples from Training Dataset

The data is preprocessed and then split into two, where the training dataset contains 50,000 images and the test set contains 10,000 images. Furthermore, three samples from each category are visualized.



The visualization showcases 3 images from each category.

## [Q2] Keras Model Implementation

The Keras model is implemented using the WRN-16-4 Model for image classification. The model specifies 3 layers of blocks with different filter sizes. The initial convolution layer uses filter size 64, the first block group uses size 128, and the last uses size 256. The input passes through all blocks because each block learns different abstract and complex features from the data. Ultimately, the total number of trainable parameters is calculated and printed. Additional code from [C2] we calculate that the total number of trainable parameters is 2,747,082.

A screenshot of a computer

AI-generated content may be incorrect.

## 

## 

Derivation of total trainable parameters

## [Q3] Usage of Adding Gaussian Noise

Adding Gaussian Noise to the images during training can help the model learn more effectively. The noise slightly changes the images by adding random noise, which makes the model focus on important patterns in the pictures, instead of smaller details. Consequently, the model is more flexible and better at handling new, unseen images. Gaussian Noise implementation also helps mitigate potential noise in the dataset, hence improving accuracy. In C3, Gaussian Noise is implemented in line 8 using tf.keras.layers.GaussianNoise.

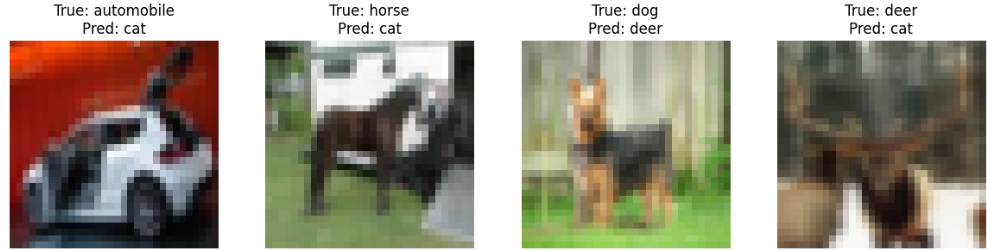
## [Q4] Model Training

The model is trained using the Wide Residual Network architecture for 20 epochs. The Adam optimizer, which is an optimization algorithm used for training deep learning models, is defined as a learning rate of 1e x . The loss for the first epoch was 1.49 (test accuracy ≈ 43%). For the last epoch, loss was equal to 0.246 (test accuracy ≈ 78%). The average loss over all 20 epochs was 0.2507. Essentially, loss measures the difference between the model’s predictions and the actual target values. With more epochs loss decreased, and accuracy increased (even though there are inconsistencies), which suggests that the model learned patterns with more epochs.

## [Q5] Model Accuracy

The classification model is tested using the evaluate\_accuracy function which calculates the accuracy on the test dataset. Furthermore, four misclassified images are identified. These examples are visualized with their true- and predicted class names. The purpose of this visualization is to understand in which cases the model struggles the most. Ultimately, the test accuracy was evaluated to 78.50%, which is moderate. There is room for improvement in this classification model.

The previous model, which evaluated model performance on train data, had an accuracy of roughly 78% on training data. This is roughly the same as accuracy on the test data which indicates that the model is not overfitting.



3 misclassified samples (along with the predicted and ground truth label).

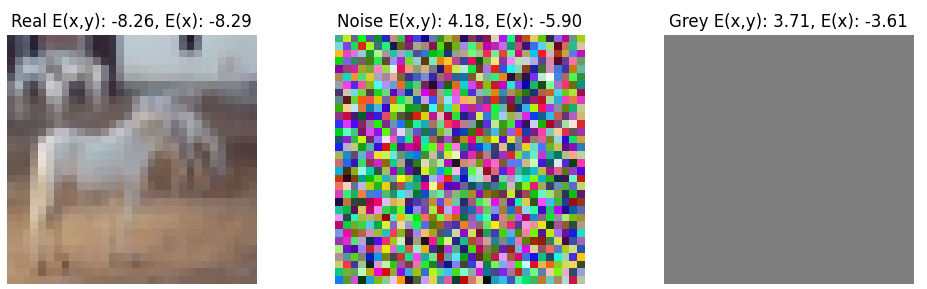
## [Q6] Accuracy of Regularized Model

After training both the original and regularized models, there is a modest improvement in test accuracy with the regularized model. The original model reached an accuracy of 78.5%, whilst the regularized model reached an accuracy of 76.5%. This suggests that the batch normalization helped reduce overfitting slightly. However, it is evident that the improvement is not drastic which suggests that the original model mitigates overfitting in the first place. The results confirm that relatively simple regularization can contribute to reliable performance improvement on the test set.

# **Part 2: Generation Task**

## [Q7] Energy-Based Evaluation of Images

The energy visualization results showcase a clear difference between real and artificial images based on energy score. The real image shows low energy values, with E(x,y) = -8.26 and E(x) = -8.29. This indicates that the model identifies the real image as a realistic example. The noise image had much higher energy scores, E(x,y) = 4.18 and E(x) = -5.9, which suggests that the model recognizes it as unrealistic input. Lastly, the grey image falls in between, with values of E(x,y) = 3.71 and E(X) = -3.61. The result indicates that the grey image is considered less likely to be realistic than the real image, but not as clearly artificial as the noise. This exercise shows how the model can effectively use energy scores to discover the likelihood that an image is real, without having labels attached to them.

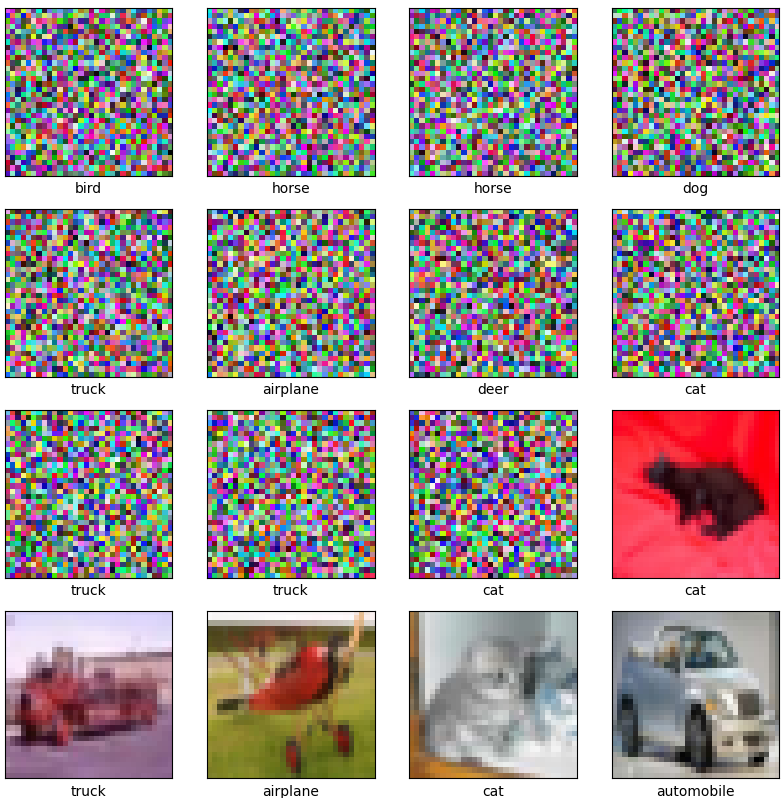


Results for real image, noise image and grey image

## [Q8] Visualization Buffer Samples

When running the visualize\_buffer\_samples function with p\_new = 0.5 there is a mix of new and stored samples. Setting p\_new to 0.5 means that the function has 50% chance of generating new data. Since the value is relatively moderate, the model will mix between exploring new parts of the data space and using stored samples. A higher value (closer to 1) would lead to a larger proportion of the samples being newly generated from random noise, vice versa. Out of 16 images, 11 were recognized as noise by the model, which suggests that they were newly generated by the model as noise and not recognized as valid images. The remaining 5 images were correctly mapped to their categories, indicating that the sampling mechanism was successfully implemented and resembled real data.

Note that the distribution and number of images that were correctly recognized, and those as noise, will vary each time the cell is run.



Visualization of buffer samples with p\_new = 0.5

## [Q9] JEM Training and Loss Analysis

The last model was trained using train\_loop\_2 for two epochs with the Join Energy-based Model (JEM) training procedure. This procedure involves classifying images and generating realistic-looking ones. It combines classifying real images and telling apart real vs. fake images by assigning energy values. This model gives low energy to real images and high energy to fake ones, which is the way it learns what real data looks like. From the first to second epoch, the loss decreased, which indicates that the model learned effectively. In the first epoch, the average loss was 1.7. In the second epoch the loss was 1.63. It is evident that adding energy loss to the cross-entropy loss makes the model learn both to classify images and to generate more realistic ones, hence reducing loss.

## [Q10] JEM Accuracy Analysis

After training the JEM model for two epochs, its accuracy was lower than the model in part 1. For the first epoch, the accuracy reached 37.21%, and the second epoch had a slight increase of 40.77%. This drop in accuracy is likely due to the increased complexity of the JEM loss and sampling steps, which make training less stable. JEM wants to classify and model data distribution, which requires more training time. Consequently, it may underperform early on, but performance can eventually improve with more training.