*Harsimran Sondhi*

*201469795*

*COMP338 – aSSIGNMENT 2*

Table of Contents

[1. Baseline Model 2](#_Toc185345491)

[1.1 CNN vs. ResNet 2](#_Toc185345492)

[2. Baseline Model Design 3](#_Toc185345493)

[2.1 Model Architecture 3](#_Toc185345494)

[2.2 Design Reasoning 3](#_Toc185345495)

[2.3 Training Configuration 3](#_Toc185345496)

[2.4 Training Results 4](#_Toc185345497)

[2.5 Observations 4](#_Toc185345498)

[3. Improved Model Design 4](#_Toc185345499)

[3.1 Enhanced Architecture 4](#_Toc185345500)

[3.2 Modifications and Reasoning 5](#_Toc185345501)

[3.3 Training Results 6](#_Toc185345502)

[3.4 Training Configuration 6](#_Toc185345503)

[4. Comparison of Results 6](#_Toc185345504)

[5. Conclusion 7](#_Toc185345505)

[6. References 7](#_Toc185345506)

# Baseline Model

## 1.1 CNN vs. ResNet

I faced the challenge of deciding whether to use a traditional Convolutional Neural Network (CNN) or a more advanced Residual Network (ResNet) as the baseline model for CIFAR-100 classification. Both models offered unique advantages and limitations, which I explored through both analysis and implementation.

CNN is a simple architecture, with fewer layers compared to ResNet. This significantly reduces its computational cost, making the CNN efficient, as it enables faster training and convergence whilst maintaining good performance. This low computational cost is ideal for datasets like CIFAR-100, where images are small (32x32), and the CNN is sufficient to extract meaningful features without overfitting.

The three convolutional layers in the CNN progressively capture hierarchical features from the images. The first layer focuses on extracting low-level features such as edges and corners, and the second layer captures intermediate features like simple shapes and textures. In addition, the third convolutional layer extracts high-level, complex features such as object parts or finer details, for accurate classification. This progression allows the CNN to effectively learn representations that are well-suited to the small resolution of CIFAR-100 images. [1] [2]

Residual Networks (ResNets) address challenges in training deep neural networks through skip connections or residual blocks, which helps remove the vanishing gradient problem and enable the effective training of much deeper architectures. This design allows gradients to flow more easily through the network, facilitating the training of networks with many layers without significant degradation in performance. [3] However, training ResNets requires significantly more computational resources and time due to their larger parameter count. Additionally, their complexity increases the risk of overfitting on smaller datasets like CIFAR-100 unless strong regularization techniques are applied. Implementing and fine-tuning ResNets can also be more challenging compared to basic CNNs. [4]

In order to choose between these two models I had to consider the balance between simplicity and performance. Despite ResNet providing deeper representational power, the computational demands and training time make it less efficient for smaller datasets like CIFAR-100. Whereas, CNN offers a simple baseline that balances efficiency and performance, providing strong results with faster training times.

I implemented both models and observed that the CNN architecture was easier to implement and train, as its reduced number of layers resulted in faster convergence in comparison to ResNet50V2, which ran double the runtime. The CNN achieved higher accuracy, delivering 44.66%, compared to 43.78% for ResNet. This significant increase in computational efficiency and accuracy demonstrates that CNN is better suited as a baseline model for CIFAR-100 classification.

# 2. Baseline Model Design

## 2.1 Model Architecture

The baseline model is a Convolutional Neural Network (CNN) consisting of three convolutional layers and a fully connected layer. The architecture is as follows:

| **Layer** | **Details** | **Output Shape** |
| --- | --- | --- |
| Input Layer | 32x32 RGB image | (32, 32, 32) |
| Conv2D | 32 filters, 3x3 kernel, ReLU | (30, 30, 32) |
| MaxPooling2D | 2x2 pool size | (15, 15, 32) |
| Conv2D | 64 filters, 3x3 kernel, ReLU | (13, 13, 64) |
| MaxPooling2D | 2x2 pool size | (6, 6, 64) |
| Conv2D | 128 filters, 3x3 kernel, ReLU | (4, 4, 128) |
| Flatten | Converts to 1D vector | (2048) |
| Dense | 256 neurons, ReLU activation | (256) |
| Dropout | Dropout rate: 0.5 | (256) |
| Dense (Output) | 100 neurons, Softmax | (100) |

## 2.2 Design Reasoning

* **Convolutional Layers**: Gradually increasing the number of filters (32 → 64 → 128) allows the model to extract low-level to high-level features.
* **Pooling Layers**: Down-sampling reduces spatial dimensions, computational cost, and overfitting.
* **Dense Layer**: The fully connected layer combines all features for final classification.
* **Dropout**: Reduces overfitting by randomly dropping neurons during training.
* **Softmax**: Outputs probabilities for 100 classes.

## 2.3 Training Configuration

* **Optimiser**: Adam with a learning rate of 0.001.
* **Loss Function**: Categorical Cross-Entropy.
* **Batch Size**: 64.
* **Epochs**: 50.
* **Data Augmentation**:
  + Random rotations: 15°
  + Horizontal flips
  + Width/height shifts: 0.1

## 2.4 Training Results

| **Metric** | **Value** |
| --- | --- |
| **Test Accuracy** | 44.66% |
| **Final Loss** | 2.1194 |

## 2.5 Observations

The baseline model achieved a **44.66%** accuracy on the CIFAR-100 test set. However:

1. **Overfitting**: A significant gap between training and validation accuracy suggests that the model is learning the training data too well, including its noise and outliers, which hampers its ability to generalise to new, unseen data. [5]
2. **Slow Convergence**: The training process was slow, due to the lack of optimisation techniques like Batch Normalisation and Learning Rate Scheduling.
   * Batch Normalisation: This technique normalises the inputs of each layer, stabilising the learning process and allowing for higher learning rates, which can lead to faster convergence. [6]
   * Learning Rate Scheduling: Adjusting the learning rate during training helps in fine-tuning the model's convergence, preventing issues like overshooting minima or getting stuck in suboptimal solutions.[6]

# 3. Improved Model Design

## 3.1 Enhanced Architecture

The improved model incorporates additional layers and techniques for better performance. The updated architecture is:

| **Layer** | **Details** | **Output Shape** |
| --- | --- | --- |
| Input Layer | 32x32 RGB image | (32, 32, 32) |
| **Block 1** | Conv2D (32 filters, 3x3) + BN | (32, 32, 32) |
|  | Conv2D (32 filters, 3x3) + BN | (32, 32, 32) |
|  | MaxPooling2D (2x2) + Dropout(0.25) | (16, 16, 32) |
| **Block 2** | Conv2D (64 filters, 3x3) + BN | (16, 16, 64) |
|  | Conv2D (64 filters, 3x3) + BN | (16, 16, 64) |
|  | MaxPooling2D (2x2) + Dropout(0.25) | (8, 8, 64) |
| **Block 3** | Conv2D (128 filters, 3x3) + BN | (8, 8, 128) |
|  | Conv2D (128 filters, 3x3) + BN | (8, 8, 128) |
|  | MaxPooling2D (2x2) + Dropout(0.4) | (4, 4, 128) |
| Flatten | Converts to 1D vector | (2048) |
| Dense | 512 neurons, ReLU + BN | (512) |
| Dropout | Dropout rate: 0.5 | (512) |
| Dense (Output) | 100 neurons, Softmax | (100) |

## 3.2 Modifications and Reasoning

**Batch Normalisation (BN)**

* **Reasoning**: BN normalises activations to reduce internal covariate shift, making training more stable and faster. [7]
* **Impact**:
  + Stabilised gradient flow and accelerated convergence.
  + Improved generalisation by reducing over-reliance on activation scales.

**Additional Conv2D Layers**

* **Reasoning**: Deeper networks can extract more hierarchical and complex features.
* **Impact**:
  + Enhanced feature extraction for CIFAR-100’s complex classes.
  + Improved model capacity without excessive parameters.

**Dropout Adjustments**

* **Reasoning**: Dropout reduces overfitting by randomly disabling neurons during training.[7]
* **Impact**:
  + Dropout rates of **0.25 and 0.4** further controlled overfitting.
  + Improved robustness and generalisation.

**Learning Rate Scheduling**

* **Reasoning**: A fixed learning rate can hinder training once the loss plateaus. ReduceLROnPlateau dynamically reduces the learning rate by **50%** when validation loss stops improving.[7]
* **Impact**:
  + Allowed fine-tuning in later epochs.
  + Improved convergence speed and final model accuracy.

**Early Stopping**

* **Reasoning**: Overfitting occurs when training continues past the point of optimal validation loss.
* **Impact**:
  + Stopped training early when validation loss plateaued.
  + Saved time and retained the best-performing model.

## 3.3 Training Results

| **Metric** | **Value** |
| --- | --- |
| **Test Accuracy** | **61.23%** |
| **Final Loss** | **1.3644** |

## 3.4 Training Configuration

Same as the baseline, but with the following:

* **ReduceLROnPlateau**: Patience = 5, Factor = 0.5.
* **EarlyStopping**: Patience = 10, restores the best weights.

# 4. Comparison of Results

| **Model** | **Test Accuracy** | **Test Loss** |
| --- | --- | --- |
| **Baseline Model** | 44.66% | 2.1194 |
| **Improved Model** | 61.23% | 1.3644 |

**Key Observations**:

* **Accuracy**: The improved model outperforms the baseline by **16.57%**.
* **Loss**: The lower validation loss indicates better generalisation.
* **Stability**: BN and learning rate scheduling stabilised training.
* **Regularisation**: Dropout adjustments and early stopping reduced overfitting.

# 5. Conclusion

The project successfully demonstrated step-by-step improvements:

1. **Baseline Model**: Achieved **44.66% accuracy** with a simple architecture.
2. **Improved Model**: Integrated Batch Normalisation, Dropout adjustments, deeper architecture, and learning rate scheduling to achieve **61.23% accuracy**.

**Findings:**

* Batch Normalisation accelerates training and improves generalisation.
* Dropout and Early Stopping effectively prevent overfitting.
* Incremental improvements and advanced training techniques significantly boost performance.

# 6. References

[1] Ultihash.io. (2024). *UltiHash object storage - Convolutional Neural Networks*. [online] Available at: https://www.ultihash.io/tech-overviews/cnns [Accessed 15 Dec. 2024].

[2] GeeksforGeeks (2024). *What are Convolution Layers?* [online] GeeksforGeeks. Available at: https://www.geeksforgeeks.org/what-are-convolution-layers [Accessed 15 Dec. 2024].

[3] Restack.io. (2024). *Resnet-18 Architecture Fine-Tuning | Restackio*. [online] Available at: https://www.restack.io/p/resnet-18-architecture-answer-fine-tuning-cat-ai [Accessed 15 Dec. 2024].

[4] Lokesh Borawar and Kaur, R. (2023). ResNet: Solving Vanishing Gradient in Deep Networks. *Lecture notes in networks and systems (Online)*, pp.235–247. doi:https://doi.org/10.1007/978-981-19-8825-7\_21.

[5] GeeksforGeeks (2024). *Validation vs. Test vs. Training Accuracy. Which One is Compared for Claiming Overfit?* [online] GeeksforGeeks. Available at: https://www.geeksforgeeks.org/validation-vs-test-vs-training-accuracy-which-one-is-compared-for-claiming-overfit [Accessed 15 Dec. 2024].

[6] GeeksforGeeks (2024). What is Batch Normalization In Deep Learning? [online] GeeksforGeeks. Available at: https://www.geeksforgeeks.org/what-is-batch-normalization-in-deep-learning [Accessed 15 Dec. 2024].

[7] Brown, J. (2024). *Complete Guide to Prevent Overfitting in Neural Networks (Part 2) - 33rd Square*. [online] 33rd Square. Available at: https://www.33rdsquare.com/complete-guide-to-prevent-overfitting-in-neural-networks-part-2/ [Accessed 15 Dec. 2024].

[8] Ioffe, S. and Szegedy, C. (2015). *Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift*. [online] arXiv.org. Available at: https://arxiv.org/abs/1502.03167.

‌

‌

‌

‌

‌

‌

‌

‌