A purple text on a black background

Description automatically generated

A black background with blue squares

Description automatically generated

**MINI PROJECT REPORT**

**CNN ARCHITECTURE MODIFICATION**

Reported to: Done by: Ahmad Atwi

Prof. Dijana Petrovska Ahmad Al-Harakeh

1. **INTRODUCTION:**

Convolutional Neural Networks (CNNs) have revolutionized the field of computer vision, offering unparalleled accuracy in tasks ranging from image classification to object detection. These deep learning models mimic the human visual system's hierarchical structure, enabling them to learn complex patterns from visual data efficiently. This report outlines the concerted effort to enhance a CNN model's accuracy, detailing the methodologies employed, findings, and conclusions drawn from the project.

CNNs consist of several layers, each designed to perform distinct yet complementary functions. The convolutional layer, the core building block, applies filters to the input data to create feature maps that highlight specific aspects of the input, such as edges or textures. Pooling layers reduce the dimensionality of these feature maps, simplifying the information without losing critical details, thereby making the network more efficient and less prone to overfitting. Activation functions like ReLU introduce non-linearity, enabling the network to learn complex patterns. Finally, dense (fully connected) layers compile the features extracted by previous layers to make predictions or classifications.

1. **OBJECTIVES :**

The overarching aim of this project was to refine a CNN's architecture to achieve superior accuracy and efficiency in processing visual imagery. The specific objectives were:

* **Architectural Adjustments**: To systematically modify the CNN's architecture by fine-tuning the arrangement and quantity of its layers, including convolutional, pooling, and dense layers.
* **Activation Function Exploration**: To assess the impact of various activation functions, namely ReLU (Rectified Linear Unit), tanh (hyperbolic tangent), and sigmoid, on the network's learning capability and performance.
* **Filter Optimization**: To explore the effects of varying the number and dimensions of filters in the convolutional layers, aiming to enhance the model's feature extraction and representation capabilities.

1. **BASELINE MODEL:**

The baseline model is a Convolutional Neural Network (CNN) model tailored for the CIFAR-10 dataset classification task. CIFAR-10, a benchmark in image classification challenges, comprises 60,000 color images (32x32 pixels) across ten distinct classes, with a split of 50,000 training and 10,000 test images. This dataset's diversity and complexity make it an ideal candidate for evaluating CNN performance enhancements.

* The baseline CNN architecture was designed with the following layers to facilitate feature learning and classification:
  1. **Feature Learning Layers:**

**•** A convolutional layer with 32 filters of size 3x3, followed by a ReLU activation function to introduce non-linearity.

• A 2x2 MaxPooling layer to reduce spatial dimensions and thus computational complexity.

• Two additional convolutional layers with 64 filters of size 3x3, each followed by a ReLU activation function.

• A second 2x2 MaxPooling layer to further compress the feature maps.

* 1. **Classification Layers:**
* A Flatten layer to convert the 2D feature maps into a 1D feature vector.
* A dense (fully connected) layer with 64 outputs and ReLU activation to interpret the features.
* A final dense layer with 10 outputs and a softmax activation function for class probability predictions.

**Model Compilation and Training**

For the model compilation:

* Adam optimizer was selected for its efficiency in handling sparse gradients on noisy problems.
* The loss function chosen was ‘categorical\_crossentropy’, suitable for multi-class classification tasks.
* The accuracy metric was employed to evaluate the model's performance during training and testing phases.
* The model underwent training over 10 epochs, a decision informed by preliminary tests to balance between underfitting and overfitting, ensuring adequate learning without excessive computation.
* The training process involved normalizing the input images to scale pixel values between 0 and 1, enhancing the model's ability to learn from the CIFAR-10 dataset effectively.

**Initial Training Results:**

For this Initial training we got a validation accuracy of 0.6988

And a training accuracy of 0.7843

A screenshot of a computer

Description automatically generated**A graph with blue and orange lines

Description automatically generated**

A close up of a black background

Description automatically generated

A screenshot of a graph

Description automatically generated**A screenshot of a computer

Description automatically generated**

1. **CNN MODIFICATIONS:**

**4.1 Testing Different Activation functions:**

Activation functions are at the heart of deep learning models, introducing non-linear properties that allow networks to learn complex patterns. In our pursuit to optimize the baseline CNN model for CIFAR-10 dataset classification, we embarked on experimenting with different activation functions to evaluate their impact on model performance. The primary goal was to identify an activation function that could improve model accuracy and learning efficiency without compromising generalizability.

The baseline model employed the ReLU (Rectified Linear Unit) activation function, known for its effectiveness in deep learning models due to its simplicity and capability to reduce the likelihood of vanishing gradients during training. However, to explore the potential for performance gains, we tested two additional activation functions:

* **Tanh (Hyperbolic Tangent):** A function that outputs values in the range [-1, 1], offering a different distribution of activation and potentially improving learning dynamics by centering the data.
* **Sigmoid**: Outputs values in the range [0, 1], which is particularly interesting for models where outputs can be interpreted as probabilities. Despite its historical use in neural networks, it’s known for being prone to vanishing gradients in deep networks.

For each activation function (ReLU, Tanh, Sigmoid), we maintained the baseline architecture unchanged except for the activation function applied in both convolutional and dense layers. This approach allowed for a controlled experiment focusing solely on the impact of activation functions. The models were trained on the CIFAR-10 training set and evaluated using the test set to ensure comparability of results.

For these 3 activation functions tested on 10 epoch we results in :

A graph of different colored lines

Description automatically generated

As we can notice ReLU results in the highest performance for the same number of epochs.

For that we started testing these activation functions on different number of epochs:

* **Sigmoid** activation function trained on20 epochs results in



A graph of a person with a blue line

Description automatically generated with medium confidence

Overfitting start being observed

* **Tanh** activation function trained on 15 results in



**SO**

* **ReLU:** Maintained as the control group, ReLU's performance served as the benchmark for comparing other activation functions. Its advantage lies in preventing vanishing gradient issues and allowing models to learn faster and more effectively.
* **Tanh**: Showed a slight decrease in performance compared to ReLU. While Tanh can be beneficial for certain types of networks due to its symmetrical output range, in this context, it did not lead to improved model accuracy.
* **Sigmoid**: Resulted in the lowest performance among the tested functions. Its susceptibility to vanishing gradients and saturation at both tails of the function likely hindered the network's learning capability.

More Training was done on different epochs but here is an example.

**4.2 Testing different kernel sizes and numbers:**

The choice of kernel size and number affects the model's view and processing of input images. Larger kernels can capture broader features with a single filter but may miss finer details, while smaller kernels can capture finer details but may require more layers to cover larger spatial hierarchies. Similarly, increasing the number of kernels allows the model to learn a more diverse set of features at the cost of computational efficiency and complexity.

We considered the following variations:

* **Kernel Sizes:** We tested kernels of sizes 1x1, 3x3, and 5x5 to compare the effects of capturing very local features versus more spatially extended features.
* **Number of Kernels:** The number of filters in each convolutional layer was varied to assess the impact on the model’s capacity to learn diverse features. We tested configurations with 32, 64, and 128 filters per layer.

Each configuration was trained on the CIFAR-10 training dataset and evaluated on the test set, maintaining consistency with other experimental conditions for accurate comparison.

**SO**

* **Kernel Sizes:**

1. 3x3 kernels struck the best balance between computational efficiency and the ability to capture relevant spatial features, confirming their widespread use in CNN architectures.
2. 5x5 kernels did not significantly enhance model performance, suggesting that the additional computational complexity was not justified given the CIFAR-10 dataset's resolution and complexity.
3. Additional training was done on a 1x1 filter size in most of the layers



This results got after training on 1 epochs using kernel size 1x1 and ReLU activation function .

We can see the drop in the accuracy

* **Number of Kernels:**

Models with 32 and 64 filters performed well, offering a good balance between learning capacity and computational demands.

Increasing to 128 filters improved the model's ability to distinguish between more nuanced features but also increased the risk of overfitting and significantly raised the computational cost.

A screen shot of a computer code

Description automatically generated

A graph of different colors

Description automatically generated

**4.3 Adding Layers :**

Adding layers to a Convolutional Neural Network (CNN) is a straightforward approach to increase its capacity for feature representation. This strategy, however, requires careful consideration to avoid overfitting and ensure the network can still generalize well to unseen data. In this phase of our project, we explored the impact of adding convolutional and pooling layers on the CNN's ability to classify images from the CIFAR-10 dataset more accurately.

To systematically assess the impact of adding layers, we experimented with incrementally introducing additional convolutional and MaxPooling layers to our baseline model. Each new configuration was trained using the same dataset and evaluated against the CIFAR-10 test set. Specifically, we examined the effects of:

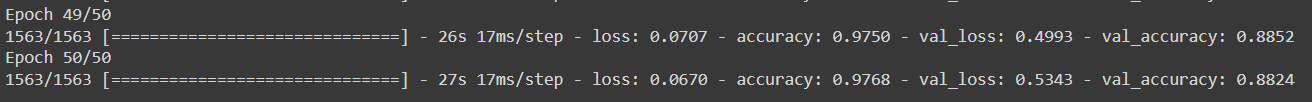
* Adding an extra convolutional layer with ReLU activation, followed by a MaxPooling layer, to the existing architecture.
* Increasing the sequence of convolutional layers before applying MaxPooling, aiming to deepen the feature extraction process without immediately reducing the spatial dimensions of the feature maps.

This results in a higher accuracy when trained on 10 epochs



Training this model on 50 epochs results in :

A graph of a curve

Description automatically generated with medium confidence

**5.Conclusion:**

This report has systematically explored various strategies to enhance the performance of a Convolutional Neural Network (CNN) for the CIFAR-10 image classification task. Starting from a baseline model, we ventured into a series of modifications aimed at improving model accuracy and generalization to unseen data. These modifications included experimenting with different activation functions, adjusting kernel sizes and numbers, adding more convolutional and pooling layers, and integrating advanced features such as batch normalization, dropout, and global average pooling.

**NOTE:**

The report contain mainly examples of trained models weren’t present in the presentation and some explanation in a detailed way .

And In order not to have a dense report we chose some examples to present rather than presenting multiple example in each case.