Shallow Convolutional Neural Network for Image Classification

Nishanth Marer Prabhu
Northeastern University, Boston, MA
marerprabhu.n@northeastern.edu

Zhijun Hu
Northeastern University, Boston, MA
hu.zhij@northeastern.edu

Tejal Patel
Northeastern University, Boston, MA
patel.tejal@northeastern.edu

Abstract

Computer vision has significantly benefited from deep learning, with the depth of neural networks being pivotal in high-accuracy achievements. However, such depth introduces substantial computational complexity. [1] explored various scaling dimensions of neural networks, but shallow networks with increased width and resolution have been less studied. This work investigates shallow neural networks' potential to deliver high performance by scaling horizontally rather than vertically.
Code: Github Repository

Software

- ***PyTorch***[2]: An open-source machine learning library based on the Torch library, used for applications such as computer vision, natural language processing, and deep learning.
- ***Simple Linux Utility for Resource Management*** [3]: An open-source job scheduler for Linux and Unix-like kernels on computer clusters.

1 Introduction

Computer vision has significantly benefited from deep learning, with the depth of neural networks being pivotal in high-accuracy achievements. However, such depth introduces substantial computational complexity. Tan & Le (2019) [4] explored various scaling dimensions of neural networks, but shallow networks with increased width and resolution have yet to be studied. This work investigates shallow neural networks' potential to deliver high performance by scaling horizontally rather than vertically.

Deep learning has profoundly transformed the field of computer vision, primarily by deploying deep neural network architectures that achieve remarkable accuracy across a multitude of tasks. While the benefits of deep networks are undeniable, their architectural complexity often results in substantial computational demands and efficiency concerns, particularly in resource-constrained environments. This study explores an alternative approach by investigating the potential of shallow neural networks that expand horizontally—by increasing width and resolution—while maintaining minimal depth. Our research centers around implementing a custom architecture, ShallowNet, which employs multiple parallel processing streams to potentially achieve competitive performance without the extensive depth typical of conventional deep learning models.

2 Methodology

2.1 Problem Formulation

The primary problem addressed in this research is the capability of shallow neural networks to match or exceed the performance of deeper networks in computer vision tasks by scaling horizontally. Understanding 2D convolution,

network depth, width, and resolution is crucial to this investigation. This study tests the hypothesis using the ParNet architecture [5], which keeps depth shallow while scaling other dimensions and incorporating parallel processing streams for performance enhancement.

The central challenge addressed in this research is the effectiveness of shallow neural networks in performing complex computer vision tasks as competently as their deeper counterparts. Deep neural networks, like ResNet and VGG, typically increase their depth with additional layers to enhance learning capability, directly translating into higher computational overhead. This research hypothesizes that a network with increased width and resolution but restricted depth can offer a viable alternative, achieving high accuracy while potentially reducing computational expense and complexity.

2.2 Model Architecture

2.2.1 Deep Neural Network

Deep Neural Networks (DNNs) are artificial neural networks (ANNs) characterized by multiple layers between the input and output layers. Each layer consists of interconnected nodes, also known as neurons, and each connection between nodes is associated with a weight. DNNs can learn complex patterns and representations from data, making them powerful tools for tasks such as image and speech recognition, natural language processing, and more.

2.2.2 Visual Geometry Group (VGG-Net) Model

Basic Building Blocks: The VGG architecture as seen in Figure 1 is composed of two repeated blocks of convolutional layers. Each convolutional layer uses a 3x3 filter with a stride of 1 and a padding of 1, which helps to preserve spatial resolution while extracting features. Each CNN layer is followed by batch normalization and an activation function.

Fully Connected Layers: Towards the end of the network, several fully connected layers are followed by a softmax activation function for classification tasks. These layers combine the learned features from earlier layers to make predictions.

ReLU Activation: Rectified Linear Unit (ReLU) activation functions are used after each convolutional and fully connected layer to introduce non-linearity into the model and allow it to learn complex patterns in the data.

Dropout: Dropout layers are often included after the fully connected layers to prevent overfitting by randomly dropping a fraction of the neurons during training.

2.2.3 Residual Network (ResNet) Model

Residual Blocks: The fundamental building blocks of the ResNet model are residual blocks, as seen in Figure 2. A residual block contains skip connections (also known as shortcut connections) that allow the network to skip one or more layers, enabling the propagation of gradients more effectively during training. These skip connections mitigate the vanishing gradient problem, which is a common issue in training very deep neural networks.

Identity Mapping: The skip connections in ResNet allow the input to bypass one or more layers and be directly added to the output of deeper layers. This forms a residual connection, which adds the original input to the output of the stacked layers.

Mathematically, the output of a residual block H(x) is defined by equation 1, and its derivative is in equation 2.

$$H(x) = F(x) + x \tag{1}$$

$$\frac{dH(x)}{dx} = \frac{dF(x)}{dx} + 1 \tag{2}$$

Basic Architecture: ResNet's basic architecture typically consists of convolutional layers. Each block may contain two or more convolutional layers with batch normalization and ReLU activation functions. The skip connections are introduced through the addition operation.

2.3 Network Architecture

The Shallow architecture is implemented with a focus on increasing network width and input resolution while maintaining a constant shallow depth. Experiments are conducted using standard datasets like CIFAR10. Performance is

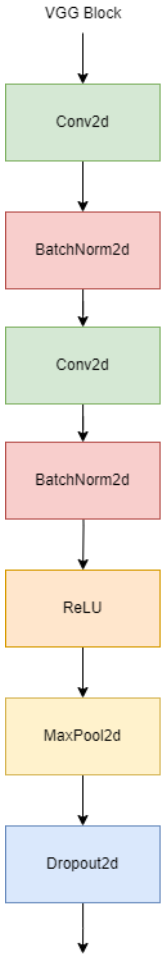

Figure 1: VGG Block

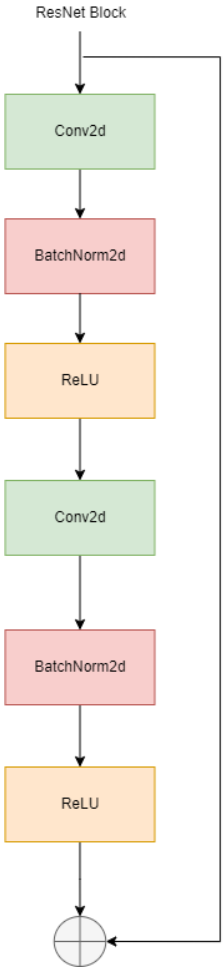

Figure 2: ResNet Block

benchmarked against traditional deep network architectures such as VGGNet, which has 5 layers of CNN and 3 fully connected layers, with a primary evaluation based on accuracy.

Our proposed ShallowNet architecture is structured as a shallow, wide network incorporating three parallel streams, each consisting of tailored convolutional blocks. This design is intended to maximize the network's ability to process and classify images efficiently at a reduced depth.

2.4 Depth Wise Separable Convolution

Depth-wise convolution decomposes the standard convolution operation into two separate operations refer to Figure 3: depth-wise convolution and point-wise convolution. Depth-wise convolution applies a single convolutional filter per input channel, while point-wise convolution applies a 1x1 convolution to combine the output channels of the depth-wise convolution. This approach reduces the computational cost and number of parameters compared to traditional convolutional layers while preserving the representative capacity of the model.

To understand this much better, let us observe the number of computations required for performing the depth-wise separable convolution compared to the regular convolution.

Assume we have an input image of size $(H * W * D) = (8 * 8 * 3)$, where H is the height, W is the width, and D is the number of channels. Let us say that post-convolution, we want to maintain the image size; however, we want to increase the number of channels from 3 to 128. Assume a filter size of $5 * 5$ is used.

The equation for determining the number of multiplications is given by,

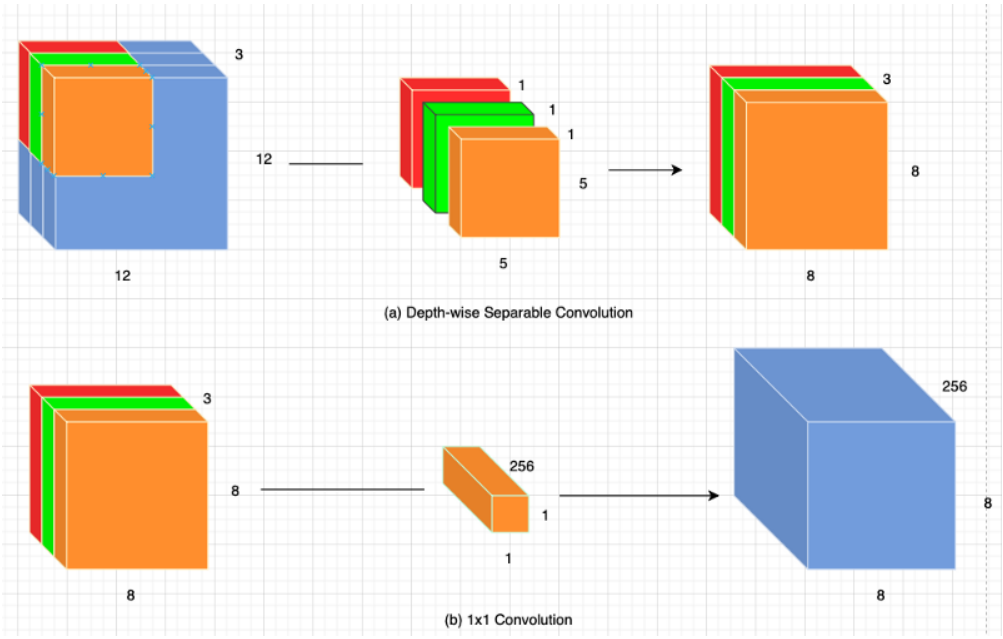

Figure 3: Depth wise Separable Convolution

$$Computation = Image_Size * (inputFiltersize) * (NumberofOutputchannels) \tag{3}$$

The input filter size corresponds to $(H * W * D) = (5 * 5 * 3)$, number of output channels is 128. Therefore, substituting in values in equation 3, we get,

$$Computation = 8 * 8 * (5 * 5 * 3) * 128 = 614400 \tag{4}$$

Now, let us apply for the depth-wise separable convolution,

The equation for determining the number of multiplications for depth-wise convolution is given by,

$$Computation = Image_Size * (inputFiltersize) * (NumberofInputchannels) \tag{5}$$

Observe that, in this case, each channel in the input image gets applied with a single filter only. Hence, the number of input channels equals the number of output channels.

Substituting in equation 5, we get the first half of the computation expense.

$$Computation = 8 * 8 * (5 * 5 * 1) * (3) = 4800 \tag{6}$$

For the second part, where we perform the point-wise convolution, the same equation 5 holds; however, in this case, the filter size is $(1 * 1)$, and the number of output channels equals the desired output channels.

$$Computation = 8 * 8 * (1 * 1 * 3) * (128) = 24576 \tag{7}$$

Combining equation 6 + 7, we get $4800 + 24576 = 29376$.

If we compare the number of computations required, we obtain a significant drop from 614400 to 29376.

2.5 Our Implementation

Let us look at the implementation of ShallowNet. The network diverges from a traditional deep layer to have multiple streams. Each stream is designed to process the input at a different resolution. The network has been implemented to use VGG or ResNet blocks as their basic blocks for performing the convolution on the dataset.

We have implemented 12 different models, all with slight architectural variations. Each model is modified slightly to assess any performance variation. Based on the reference paper [5], which states that the number of streams is set to 3, we try to create a model with a higher number of streams to observe any performance gain or drop.

Below, we explain the basic architecture on which all the models are based.

1. Main Stream Configuration: Begins with a basic convolutional layer setup that processes the input image to prepare it for detailed feature extraction in subsequent streams. This stream uses two convolutional layers with stride adjustments, including max pooling and dropout layers, to prevent overfitting.
2. Parallel Stream Architectures: Each of the three streams diverges from the main flow, applying convolutional layers of varying depths. Constant filter size is applied across all convolutional layers to capture features at different scales and complexities. Post-performing convolution in each parallel stream, each stream reduces the image's resolution by half to ensure concatenation with the other streams. This may be done before or after the fusion blocks.
   (a) Stream 1 and Stream 2: Focus on intermediate features, using sequences of convolutional and batch normalization layers to deepen the feature maps before they are merged.
   (b) Stream 3: Targets deeper feature extraction with a similar configuration, ensuring comprehensive feature analysis across the network.
3. Fusion and Pooling: The outputs of the three streams are concatenated and subsequently pooled to synthesize and compact the feature maps, preparing them for classification.
4. Classification Block: Concludes with fully connected layers that map the extracted features to the ten class labels. This block also includes dropout layers and a softmax activation function to output class probabilities.

2.5.1 VGGBlock Shallow Net

Referring to the section 2.5, Figure 4 represents the architecture of the Shallow Net, which uses the VGG block as its basic block. This network represents our simplest network, which has only 4 layers of CNN and 3 layers of Fully connected layers, and we call this "ShallowNet 3 Streams 1 Block". It contains three streams, each with a VGG block represented in grey.

Similar to this model configuration, we have created two more models, namely:

1. "ShallowNet 3 streams 2 blocks": The model consists of three parallel streams containing two VGGBlocks.
2. "ShallowNet 4 streams 2 blocks": The model consists of four parallel streams, each containing two VGGBlocks.

Furthermore, to reduce the number of trainable parameters in the network and aim to reduce the model size, we utilized the concept of depth-wise separable convolution 2.4 and created a model namely:

1. "ShallowNet 3 streams 1 block Depth": The model consists of three parallel streams, and each stream contains one VGGBlock and uses depth-wise separable convolution.

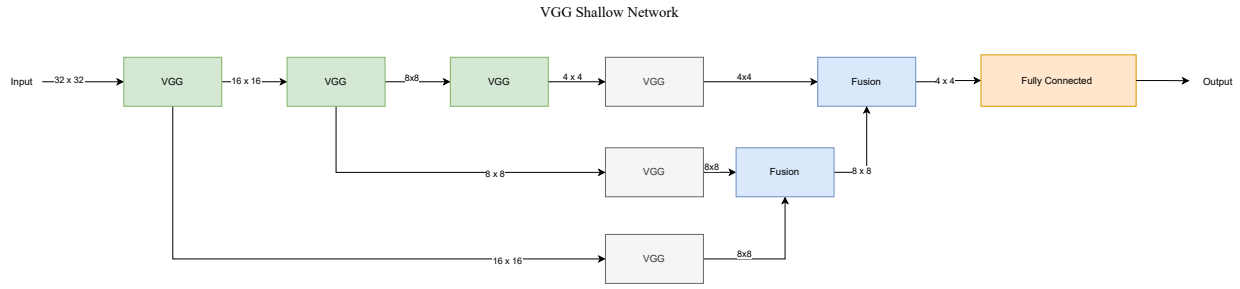

Figure 4: Shallow Net using VGG Blocks

2.5.2 ResNetBlock Shallow Net

Similar to the VGGBlock implementation, to analyze the effect of changing the basic block, we try to utilize the ResNet block for our shallow network implementation. The basic design and architecture remain the same. Figure 5 represents the simple version of the ShallowNet 3 Stream 1 Block ResNet model.

Similar to this model, we have created a variation to understand the residual connection's benefits.

1. "ShallowModel ResNet 3 Streams 1 Block": The model consists of three parallel streams, each containing one ResNet Block.
2. "ShallowModel ResNet 3 Streams 2 Block": The model consists of three parallel streams. Each stream contains two ResNet Blocks.
3. "ShallowModel ResNet 3 Streams 3 Block": The model consists of three parallel streams. Each stream contains three ResNet Blocks.

We also utilized the depth-wise convolution 2.4 for these models and created three variants. Furthermore, we have created two models that contain a skip connection along the streams, i.e., if there are two blocks in a stream, then a skip connection originates before the start of the stream and connects at the end of the stream before the fusion block.

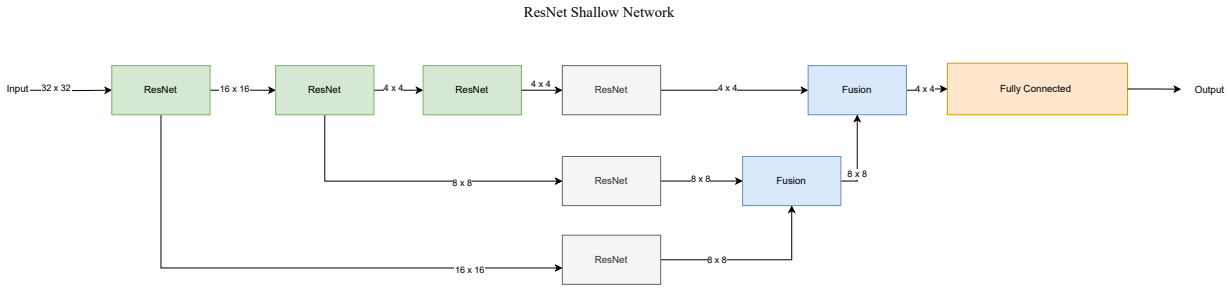

Figure 5: Shallow Net using ResNet Blocks

3 Experiments

To determine ShallowNet's benefits, we trained its multiple variations on the CIFAR10 dataset, an image dataset consisting of 10 classes. We empirically show the model's accuracy improvement compared to the baseline Deep Neural networks.

3.1 Configuration Details

All experiments were run on the Northeastern University Discover cluster using the slurm job scheduler. The following sbatch configurations at Table 1 were set for each experiment. All the sbatch scripts were submitted parallel to ensure the model training occurred seamlessly. After training each model, the models are checkpointed, which can be utilized to test the model further with individual images.

Partition	**Memory**	**Cpus-per-task**	**GPU**	**Time**
GPU	64Gb	8	P100	08:00:00

Table 1: SBATCH Configurations

We ensured consistency for all the trained and tested models by setting the same hyperparameters. In the below table 2, we use the SGD as the optimizer and the scheduler for varying the learning rate throughout the training. The scheduler ensures a higher learning rate at the beginning of training for faster convergence and slowly starts reducing it to prevent overshoot from the minimum function value.

3.2 Datasets

We have utilized the CIFAR10 dataset, which consists of 60000 32x32 color images in 10 classes, with 6000 images per class. There are 50000 training images and 10000 test images. The smaller image size makes it beneficial to prove that shallow networks are feasible and can be extended for datasets with higher image resolution.

Hyper parameter	Value
DropOut	0.1
Learning Rate	0.001
Weight Decay	0.0001
Patience	8
Batch Size	128
Epoch	200
Optimizer	SGD
Scheduler	OneCycleLR

Table 2: Hyperparameter Settings for Model Training and Testing

Dataset	Dimension (d)	Samples (N)	Classes
CIFAR10	32x32x3	60000	10

Table 3: Datasets found from [6] (CIFAR10)

3.3 Model Configurations

We have used VGGNet and ResNet9 models as our baseline estimators. These models provide a good baseline for comparing model performance in terms of accuracy and number of trainable parameters (model size).

Model Name	No. of Layers	No. of Parameters (Million)	Block Type
VGGNet	7	1.852	VGG
ResNet9	9	6.575	ResNet

Table 4: Baseline Deep Layer Models

There are twelve variations of Shallow Networks, each of which uses the VGG or ResNet block as its base as shown in the tables 5 and 6. Following this, there are models with varying numbers of streams and different numbers of blocks in each stream. Some models also utilize the depth-wise separable convolution to reduce the number of computations.

The table 5 lists the models using VGGBlock as their base, describing the number of streams, number of blocks in each stream, parameter count, and whether the network uses depth-wise convolution.

Model Name	No. of Streams	No. of Blocks in each Stream	No. of Parameters (Million)	Depth Wise CNN
ShallowModel 3Streams 1Block	3	1	1.345	No
ShallowModel 3Streams 1Block Depth	3	1	0.746	Yes
ShallowModel 3Streams 2Block	3	2	1.734	No
ShallowModel 4Streams 2Block	4	2	7.083	No

Table 5: Variations of Shallow Net Models

The table 6 lists the models using ResNetBlock as their base, describing the number of streams, number of blocks in each stream, parameter count, and whether the network uses depth-wise convolution.

Model Name	No. of Streams	No. of Blocks in each Stream	No. of Parameters (Million)	Depth Wise CNN
ShallowModel ResNet 3Streams 1Block	3	1	1.358	No
ShallowModel ResNet 3Streams 1Block Depth	3	1	0.759	Yes
ShallowModel ResNet 3Streams 2Block	3	2	1.747	No
ShallowModel ResNet 3Streams 2Block Depth	3	2	0.808	Yes
ShallowModel ResNet 3Streams 2Block Depth Skip Connection	3	2	0.808	Yes
ShallowModel ResNet 3Streams 3Block	3	3	2.135	No
ShallowModel ResNet 3Streams 3Block Depth	3	3	0.857	Yes
ShallowModel ResNet 3Streams 3Block Depth Skip Connection	3	3	0.857	Yes

Table 6: Variations of Shallow Net Models

4 Results

We empirically show that the shallow network provides a significant advantage in prediction accuracy and reduction in the model size. We see a jump of 4 to 5 percent in the testing accuracy and a reduction of 5x times in the number of trainable parameters.

4.1 Model Performance

We trained all 12 models and the two baseline models on the CIFAR10 dataset. To improve the model performance and aid the model learning ability, we performed specific data augmentation techniques like rotation, scaling, flipping, cropping, and image resizing to ensure that the models can learn and extract more from the images. (The "DataLoader 2.py" file performs additional normalization to the images compared to the DataLoader.py file, improving the model's learning ability by 2 percent.)

4.1.1 Testing Accuracy Figures for VGGNet and Shallow Net with VGG Block

As mentioned in section 3.1, all models were trained up to 200 epochs. The baseline model figures are as follows:

Model Name	Testing Accuracy (in Percentage)
VGGNet	88.2

Table 7: Baseline Model's Testing Accuracy Figures

Let's examine the testing accuracy figures in table 8 for the Shallow Nets with VGG Block as their basic block.

Model Name	Testing Accuracy (in Percentage)
ShallowModel 3Streams 1Block	**89.53**
ShallowModel 3Streams 1Block Depth	85.53
ShallowModel 3Streams 2Block	89.05
ShallowModel 4Streams 2Block	84.87

Table 8: Shallow Net Model's Testing Accuracy Figures

Comparing the figures of the baseline VGG network and the Shallow Nets, we see a slight improvement in the testing accuracy. However, the model size is reduced by 1.3x times considering the 3 streams and 1 block or 2 block configuration, which uses regular or depth-wise separable convolution.

When we observe the Shallow Net model, which consists of 4 streams, we see a drop in accuracy and a significant jump in the number of trainable parameters. This may be a case where performing the computation on a resolution too small, in this case, a 2x2 image size, may not be beneficial and does not add an improvement but instead leads to degradation of the model's performance or, in other words, the information retrieved or extracted may not be significant enough.

4.1.2 Testing Accuracy Figures for ResNet and Shallow Net with ResNet Block

The baseline figure for the ResNet9 model is as follows:

Model Name	**Testing Accuracy (in Percentage)**
ResNet9	85.0

Table 9: Baseline Model's Testing Accuracy Figures

Now, we will analyze the testing accuracy figures for the Shallow Net models with the ResNet as their basic block,

Model Name	**Testing Accuracy (in Percentage)**
ShallowModel ResNet 3Streams 1Block	**89.06**
ShallowModel ResNet 3Streams 1Block Depth	87.27
ShallowModel ResNet 3Streams 2Block	**89.18**
ShallowModel ResNet 3Streams 2Block Depth	85.62
ShallowModel ResNet 3Streams 2Block Depth Skip Connection	87.06
ShallowModel ResNet 3Streams 3Block	**89.15**
ShallowModel ResNet 3Streams 3Block Depth	85.90
ShallowModel ResNet 3Streams 3Block Depth Skip Connection	85.34

Table 10: Shallow Net Model's Testing Accuracy Figures

From the tables 9 and 10, we can observe a significant improvement in the performance figures across various models. Specifically, the highlighted models have an accuracy close to 89.2 %. Furthermore, these models have a significantly reduced number of trainable parameters in comparison to the ResNet9 model.

Though there are claims where the ResNet9 can reach up to 90 % accuracy, our testing using the published information enabled us to reach up to the values shown in table 9. Even if there is a case where it may exceed performance, the reduced parameter count makes the Shallow Net more desirable.

4.2 Performance Analysis

For in-depth analysis, let us try to understand the concept of gradient descent, specifically the partial derivatives of each weight with respect to the output during the backpropagation process.

In a regular deep-layer network, the gradient propagates through each layer, and the weights in each layer depend on the next layer; i.e., layer 1 weights depend on layer 2 derivatives. Based on those values, the weights in the upper end of the network get modified based on the variation in the lower end; refer to Figure 6.

However, observing the Figure 4, we observe a slight difference, the blocks marked in gray, which process the images at different resolutions, their weight update does not depend on the proceeding CNN layer, which performs convolution in lower resolution in other words, the block in stream 1 processes the image at 16x16, while the block in stream 2 process the image at 8x8 and the weights of these two blocks are independent of each other. This weight independence may contribute to enabling the Shallow Net to be lighter and have higher accuracy in image recognition.

For better understanding, we will use Figure 7 as a reference; in this case, we assume that neurons 3 and 4 are processing images at different resolutions. A typical neural network would have a cross-connection between them, as

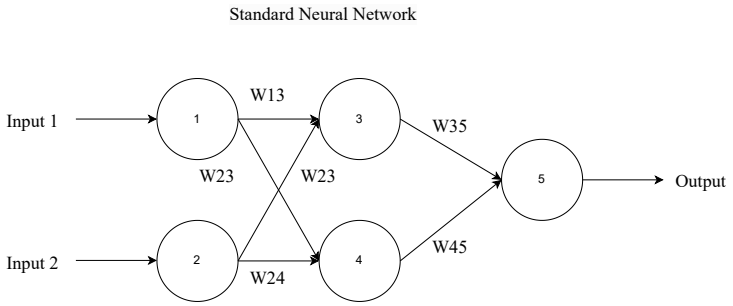

Figure 6: Shallow Net using ResNet Blocks

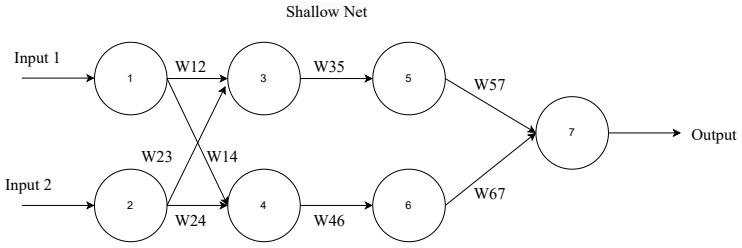

Figure 7: Shallow Net using ResNet Blocks

seen in Figure 6. The absence of this connection makes the neurons 3 and 4 independent of the partial derivatives from the neurons 5 and 6. This independence may lead to better weight adjustment in the filter kernels and improved accuracy figures.

Due to time constraints, we were unable to prove this mathematically. However, the above statement does prove to be a hypothesis worth testing. If enabling independent streams is beneficial for model performance, this could branch out into a new research stream focusing on condensing neural networks, allowing them to be trained and inferred on GPUs with lower memory bandwidth.

5 Conclusion

This work presents a comprehensive study of the Shallow Network and compares its performance with traditional deep neural networks. Through empirical evidence, we have shown that the model, in some cases, can match the performance of the standard neural network while having a reduced trainable parameter count. In other instances, we experience a significant improvement in the performance while having a significantly reduced trainable parameter count.

The Shallow Network does present some advantages in terms of accuracy and reduced model size, which enables models to be trained and deployed on GPUs with memory limitations. Furthermore, this opens up a new avenue for exploring the merits of widening a network and performing the processing in parallel, thus leading to a significant reduction in the model depth.

In conclusion, a percentage improvement of 4 to 5 % in the case of the ResNetBlock shallow network and a 5x times reduction in parameter size makes the Shallow Network more desirable as it can achieve similar performance. This is mainly useful when dealing with data with higher resolutions; in those cases, a shallow network would be preferred to a deeper one.

Future work involves using trainable activation functions to improve the network's accuracy and researching mathematically how the shallow network presents the merits of performing computations at various resolutions in parallel streams.

References

[1] M. Tan and Q. V. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks," 2020.

[2] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in pytorch," in *NIPS-W*, 2017.

[3] M. Jette, C. Dunlap, J. Garlick, and M. Grondona, "Slurm: Simple linux utility for resource management,"

[4] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," 2015.

[5] A. Goyal, A. Bochkovskiy, J. Deng, and V. Koltun, "Non-deep networks," 2021.

[6] A. Krizhevsky, "Learning multiple layers of features from tiny images," tech. rep., 2009.