Name: Freya(Wenbin) Li
Computing ID: wla202
Student ID: 301563887

Kaggle Name: LIFREYA

Kaggle Team: Siyu&Freya&Haoran

Project2 Report

Kaggle Team: Siyu&Freya&Haoran
Team members: Siyu An, Freya Li, Haoran Ding

Part1

Best accuracy on Kaggle: 0.734

1. Introduction

By introducing "shortcut" connections that bypass one or more layers, ResNet effectively addresses the vanishing gradient problem in deep neural networks. This report presents a customized implementation of the ResNet architecture, elucidating its design, improvements, and ablation studies.

2. Network Architecture

Layer/Component	Description	Kernel size	Stride	Input Channels	Output Channels
BaseNet					
Conv1	Conv+Bn+ReLU	3*3	1	3	64
Conv2	Conv+Bn+ReLU	3*3	1	64	128
Conv3	Conv+Bn+ReLU	3*3	1	128	256
MaxPool	Max pooling	2*2	2	256	256
ResBlock1					
Conv1	Conv+Bn+ReLU	3*3	1	256	256
Conv2	Conv+Bn+ReLU	3*3	1	256	256
ResBlock2					
Conv1 (downsample)	Conv(1*1)	1*1	2	256	512
Conv1	Conv+Bn+ReLU	3*3	1	256	512
Conv2	Conv+Bn+ReLU	3*3	1	512	512
ResBlock3					
Conv1 (downsample)	Conv(1*1)	1*1	2	512	1024
Conv1	Conv+Bn+ReLU	3*3	1	512	1024
Conv2	Conv+Bn+ReLU	3*3	1	1024	1024

AdaptiveAvgPool		-	-	1024	1024
Flatten	Flatten Layer	-	-	-	-
FC1	Fully Connected	-	-	1024	128
FC2	Fully Connected	-	-	128	TOTAL_CLASSES
BN	Batch Normalization(1D)			TOTAL_CLASSES	TOTAL

3. Results

```
[88] loss: 0.033
Accuracy of the network on the val images: 72 %
[89] loss: 0.035
Accuracy of the network on the val images: 72 %
[90] loss: 0.037
Accuracy of the network on the val images: 72 %
[91] loss: 0.034
Accuracy of the network on the val images: 72 %
[92] loss: 0.031
Accuracy of the network on the val images: 73 %
[93] loss: 0.031
Accuracy of the network on the val images: 72 %
[94] loss: 0.028
Accuracy of the network on the val images: 73 %
[95] loss: 0.030
Accuracy of the network on the val images: 72 %
[96] loss: 0.032
Accuracy of the network on the val images: 72 %
[97] loss: 0.028
Accuracy of the network on the val images: 73 %
[98] loss: 0.026
Accuracy of the network on the val images: 72 %
[99] loss: 0.030
Accuracy of the network on the val images: 73 %
[100] loss: 0.026
Accuracy of the network on the val images: 72 %
Finished Training
```

Figure 1

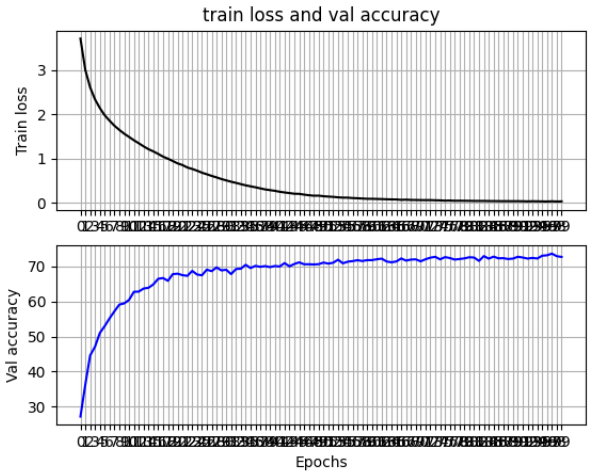

Figure 2

From the training log and the plot:

- The training loss consistently decreases over a span of 100 epochs, which is a good indication that the model is learning and fitting to the training data.

- The validation accuracy reached a peak of approximately 73%, indicating the model's ability to generalize well on unseen data, also matches our best result on Kaggle for the validation score of 0.734. However, beyond a certain number of epochs, while the training loss continued to decrease, the validation accuracy plateaued. This suggests that the model might be starting to overfit the training data after these epochs since the training loss continues to decrease, but the validation accuracy does not show significant improvement. The attached plot visually represents the training loss and validation accuracy trends over the epochs.

4. Model breakdown

Initial Convolution Layer

The model starts with a series of convolutional layers called the BaseNet. These layers set the stage for feature extraction:

A. Conv1: The first convolutional layer (Conv1) uses a 3x3 kernel and processes the 3-channel input (RGB) to produce 64 channels. This transformation is followed by Batch Normalization and ReLU activation.
B. Conv2: The subsequent layer (Conv2) expands the channel depth from 64 to 128 using a 3x3 kernel, complemented by Batch Normalization and ReLU activation.
C. Conv3: Similarly, the third convolutional layer (Conv3) elevates the depth from 128 channels to 256, again employing a 3x3 kernel with subsequent Batch Normalization and ReLU activation.
D. MaxPool: After these initial transformations, a Max Pooling layer reduces the spatial dimensions of the feature maps by half (stride = 2), taking them from 32x32 to 16x16, without altering the channel depth.

Residual Blocks

Central to the architecture are the Residual Blocks, which allow for deeper networks without the vanishing gradient problem:

A. ResBlock: This block doesn't change the channel depth. It retains 256 channels throughout. Each ResBlock contains two main convolutional layers, both using 3x3 kernels, Batch Normalization, and ReLU activation.

B. ResBlock 2: This block transitions the depth from 256 channels to 512. The block starts with a downsample layer using a 1x1 convolution to adjust the spatial dimensions and channel depth. The following convolutional layers in the block use 3x3 kernels.

C. ResBlock 3: This block further expands from 512 channels to 1024 channels. Like ResBlock 2, it starts with a 1x1 convolution for downsampling. The remaining layers in this block employ 3x3 kernels.

Each ResBlock leverages shortcut connections to add the output of the block with its input, ensuring that the network can learn residual functions.

Classification Layers

After processing through the residual blocks:

A. AdaptiveAvgPool: The feature maps are subjected to an Adaptive Average Pooling layer, which reduces the spatial dimensions to 1x1, essentially generating a feature vector of length 1024 for each image.

B. Flatten: The 2D feature maps are then flattened to form a 1D vector.

- FC1: This feature vector is passed through a Fully Connected (FC) layer, which reduces its dimensions from 1024 to 128.

- FC2: The subsequent FC layer outputs class scores corresponding to `TOTAL_CLASSES`.

C. Batch norm: Finally, a 1D Batch Normalization layer is applied to stabilize and scale the outputs.

Our architecture leverages the strengths of ResNet, especially the residual connections, to learn from deeper networks without facing issues like the vanishing gradient.

The initial convolutional layers help in capturing basic features, which are then refined and expanded upon by customized ResBlocks. The classification layers then map these refined features to class scores.

4. Ablation Study

The ablation study involves systematically removing parts of the model to understand the importance and contribution of each part to the model's performance.

A. ReLU as activation functions
Here, the performance of a modified model (without any ReLU activation functions) is compared to the performance of a complete model (which achieved 73% validation accuracy).

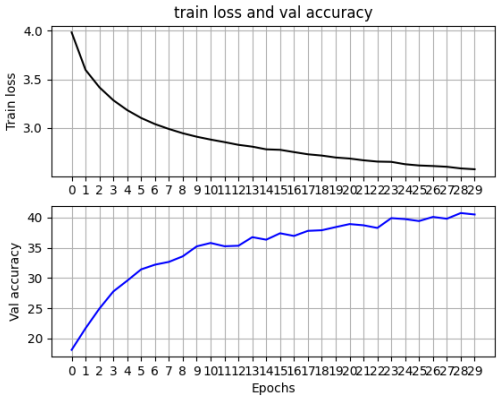

Figure 3

The modified model, lacking ReLU activation functions, reached an accuracy of merely around 40% after 30 epochs (Figure 3). ***This contrasts with our benchmark model that scaled up to an accuracy ranging between 66% to 70% within the same epoch range***. Furthermore, the rate of decline in training loss for the modified model was much slower.

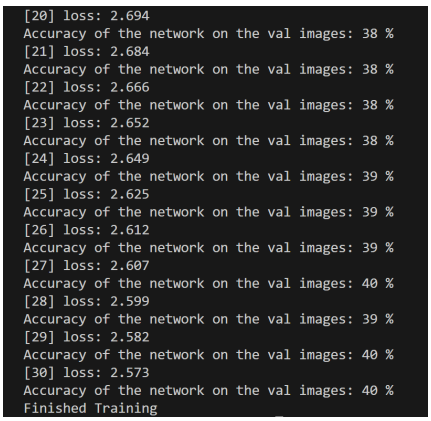

Figure 4

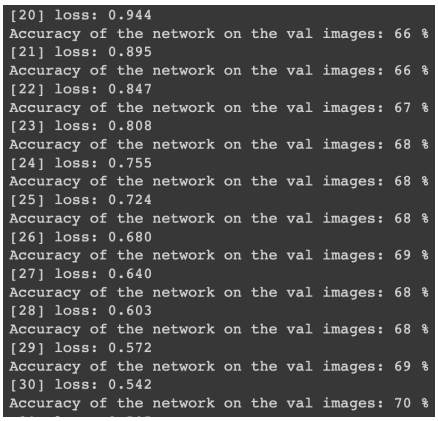

Figure 5

*Figure 4 is the validation accuracy obtained from the editing without ReLU activations.
*Figure 5 is the validation accuracy obtained from the benchmark training.

A few impacts of non-linear activations on model performance:

Non-linearity

Activation functions, especially ReLU, introduce non-linearity into the neural network. This non-linearity is significant as it makes the neural network to capture intricate patterns in the data. In the absence of such activation functions, the neural network is mostly linear, vastly limiting its modeling capacity and diminishing its performance.

Saturation

ReLU and its variants are popular for addressing the vanishing gradient problem. Without ReLU, the network might be more susceptible to this problem, especially in deeper architectures, leading to hindered learning process.

Convergence Speed

Activation functions like ReLU can potentially speed up the convergence during training. The slower descent in training loss observed in the model without ReLU might be indicative of this hindered convergence rate.

B. Normalization

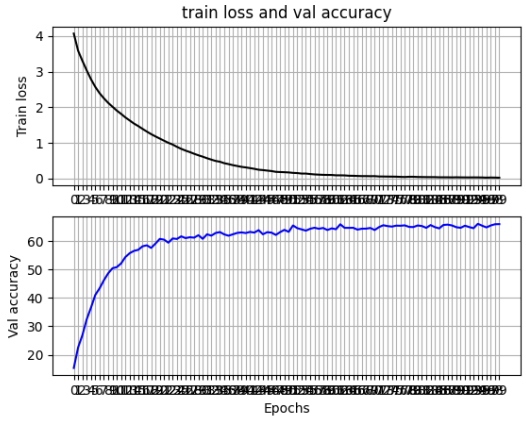

Figure 6

```
Accuracy of the network on the val images: 65 %
[88] loss: 0.038
Accuracy of the network on the val images: 65 %
[89] loss: 0.036
Accuracy of the network on the val images: 65 %
[90] loss: 0.038
Accuracy of the network on the val images: 64 %
[91] loss: 0.036
Accuracy of the network on the val images: 64 %
[92] loss: 0.035
Accuracy of the network on the val images: 65 %
[93] loss: 0.034
Accuracy of the network on the val images: 64 %
[94] loss: 0.035
Accuracy of the network on the val images: 64 %
[95] loss: 0.035
Accuracy of the network on the val images: 66 %
[96] loss: 0.033
Accuracy of the network on the val images: 65 %
[97] loss: 0.029
Accuracy of the network on the val images: 64 %
[98] loss: 0.032
Accuracy of the network on the val images: 65 %
[99] loss: 0.030
Accuracy of the network on the val images: 65 %
[100] loss: 0.027
Accuracy of the network on the val images: 66 %
Finished Training
```

Figure 7

*The plot and training log of editing without normalization are shown in Figure 6 & Figure 7 respectively.

I removed all the data and batch normalization to study the performance impact of normalization. In order to get a fair comparison, the number of epochs is set as 100, which is the same as the number we use in the benchmark model with the highest validation accuracy. The model without any normalization achieves a validation accuracy of around 65-66%. This is lower than the 73% achieved by the complete model with batch normalization. Without batch normalization, models might be more susceptible to overfitting, especially when the training loss decreases significantly while the validation accuracy doesn't increase proportionally. In the given results, ***the training loss consistently decreases over epochs, but the validation accuracy seems to plateau around 65-66%***. This might indicate some degree of overfitting, though it's not much evident.

A few impacts of normalization on model performance:

Training Stability

Without batch normalization, models might experience unstable training due to internal covariate shift. However, the provided results show a relatively stable decrease in loss over epochs, indicating that, in this specific scenario, the model was able to train reasonably well even without BN.

Overfitting

Without batch normalization, models might be more susceptible to overfitting, especially when the training loss decreases significantly while the validation accuracy doesn't increase proportionally. In our results, the training loss consistently decreases over epochs, but the validation accuracy seems to plateau around 65-66%. This might indicate some degree of overfitting, though it's not so evident.

Generalization

Batch normalization can also help the model generalize better to unseen data. The model without batch norm might not generalize as well as the model with it.

Epoch Variability

The validation accuracy fluctuates a bit over epochs, but it remains within a narrow band (64% to 66%). This suggests that the model without batch norm might be more sensitive to the specific data it sees during training.

Overall Comparisons:

Performance Impact

The model without batch normalization achieves a validation accuracy of around 65-66%. This is lower than the 73% achieved by the complete model with batch normalization but higher than the 40% achieved by the model without ReLU activations.

5. Conclusions

This custom ResNet architecture demonstrates the potential of deep residual learning. Through iterative experimentation, the model was refined to achieve superior performance. I further studied the importance of components like Normalization and Residual Learning in deep neural networks. This might suggest that while batch normalization has a positive impact on model performance, it's not as critical as the activation functions.

Part 2

Data augmentation

The training data underwent the following transformations:

- Resized to 256x256 pixels.
- Center-cropped to 224x224 pixels.
- Random horizontal flipping.
- Normalized with mean and standard deviation. Typically, I used the mean and standard deviation for the resnet18 model trained on ImageNet, the normalization constants are:

  Mean: [0.485, 0.456, 0.406]
  Standard deviation: [0.229, 0.224, 0.225]

While the above transformations improved the model's performance, further experimentation revealed the impact of additional augmentations:

- *Using `transforms.RandomRotation(degrees = (30,60))` in the fixed extractor training significantly affected the training accuracy, preventing the model from reaching higher accuracy levels observed without this transformation.*

- *Regularization:*
  Introducing a dropout of 0.25 resulted in reduced training performance with the highest accuracy around

73-74% for the entire model training. This indicates that while dropout can be effective in preventing overfitting, excessive dropout can hinder the learning process.

Fine-tuning

Depending on configurations, I either:
- Used ResNet as a fixed feature extractor, where only the final layer was trained.
- Fine-tuned the entire ResNet model.

Hyperparameters

ResNet as a fixed feature extractor

- Batch Size: 32
- Learning Rate: 0.0005
- Epochs: 80
- Optimizer: SGD with momentum of 0.9
- ResNet Configuration: Only the last layer was fine-tuned. Set 'RESNET_LAST_ONLY = True'.

ResNet

- Batch Size: 32
- Learning Rate: 0.0005
- Epochs: 45
- Optimizer: SGD with momentum of 0.9
- ResNet Configuration: Fine-tuned the entire model.

Training & Testing Results

Fine-tuned only the last layer

1) The training accuracy began at 0.63% during the first epoch.
2) By the 80th epoch, the accuracy had increased to 85.03% (as shown in Figure 8).
3) This model achieved a testing accuracy of 42.4% with a loss of 0.0285.

```
TRAINING Epoch 45/80 Loss 0.0245 Accuracy 0.7430
TRAINING Epoch 46/80 Loss 0.0242 Accuracy 0.7430
TRAINING Epoch 47/80 Loss 0.0237 Accuracy 0.7543
TRAINING Epoch 48/80 Loss 0.0235 Accuracy 0.7590
TRAINING Epoch 49/80 Loss 0.0232 Accuracy 0.7580
TRAINING Epoch 50/80 Loss 0.0228 Accuracy 0.7743
TRAINING Epoch 51/80 Loss 0.0224 Accuracy 0.7727
TRAINING Epoch 52/80 Loss 0.0222 Accuracy 0.7833
TRAINING Epoch 53/80 Loss 0.0219 Accuracy 0.7767
TRAINING Epoch 54/80 Loss 0.0216 Accuracy 0.7853
TRAINING Epoch 55/80 Loss 0.0213 Accuracy 0.7833
TRAINING Epoch 56/80 Loss 0.0210 Accuracy 0.7953
TRAINING Epoch 57/80 Loss 0.0207 Accuracy 0.8003
TRAINING Epoch 58/80 Loss 0.0205 Accuracy 0.8010
TRAINING Epoch 59/80 Loss 0.0202 Accuracy 0.8030
TRAINING Epoch 60/80 Loss 0.0200 Accuracy 0.8100
TRAINING Epoch 61/80 Loss 0.0197 Accuracy 0.8060
TRAINING Epoch 62/80 Loss 0.0194 Accuracy 0.8120
TRAINING Epoch 63/80 Loss 0.0192 Accuracy 0.8133
TRAINING Epoch 64/80 Loss 0.0190 Accuracy 0.8233
TRAINING Epoch 65/80 Loss 0.0187 Accuracy 0.8233
TRAINING Epoch 66/80 Loss 0.0185 Accuracy 0.8247
TRAINING Epoch 67/80 Loss 0.0183 Accuracy 0.8270
TRAINING Epoch 68/80 Loss 0.0181 Accuracy 0.8263
TRAINING Epoch 69/80 Loss 0.0178 Accuracy 0.8360
TRAINING Epoch 70/80 Loss 0.0176 Accuracy 0.8350
TRAINING Epoch 71/80 Loss 0.0174 Accuracy 0.8403
TRAINING Epoch 72/80 Loss 0.0172 Accuracy 0.8443
TRAINING Epoch 73/80 Loss 0.0170 Accuracy 0.8427
TRAINING Epoch 74/80 Loss 0.0169 Accuracy 0.8390
TRAINING Epoch 75/80 Loss 0.0167 Accuracy 0.8400
TRAINING Epoch 76/80 Loss 0.0164 Accuracy 0.8487
TRAINING Epoch 77/80 Loss 0.0163 Accuracy 0.8467
TRAINING Epoch 78/80 Loss 0.0160 Accuracy 0.8490
TRAINING Epoch 79/80 Loss 0.0160 Accuracy 0.8533
TRAINING Epoch 80/80 Loss 0.0158 Accuracy 0.8503
Finished Training
```

Figure 8

Fine-tuned the entire ResNet model

1) The training accuracy started at 0.53% in the first epoch.
2) By the 45th epoch, the training accuracy had reached 99.8% (as shown in Figure 9).
3) The testing accuracy was 56.97% with a loss of 0.0577.

```
TRAINING Epoch 20/45 Loss 0.0415 Accuracy 0.9137
TRAINING Epoch 21/45 Loss 0.0389 Accuracy 0.9213
TRAINING Epoch 22/45 Loss 0.0358 Accuracy 0.9307
TRAINING Epoch 23/45 Loss 0.0336 Accuracy 0.9397
TRAINING Epoch 24/45 Loss 0.0311 Accuracy 0.9447
TRAINING Epoch 25/45 Loss 0.0286 Accuracy 0.9610
TRAINING Epoch 26/45 Loss 0.0270 Accuracy 0.9617
TRAINING Epoch 27/45 Loss 0.0249 Accuracy 0.9703
TRAINING Epoch 28/45 Loss 0.0232 Accuracy 0.9750
TRAINING Epoch 29/45 Loss 0.0217 Accuracy 0.9737
TRAINING Epoch 30/45 Loss 0.0199 Accuracy 0.9830
TRAINING Epoch 31/45 Loss 0.0189 Accuracy 0.9863
TRAINING Epoch 32/45 Loss 0.0175 Accuracy 0.9870
TRAINING Epoch 33/45 Loss 0.0165 Accuracy 0.9877
TRAINING Epoch 34/45 Loss 0.0152 Accuracy 0.9913
TRAINING Epoch 35/45 Loss 0.0142 Accuracy 0.9947
TRAINING Epoch 36/45 Loss 0.0134 Accuracy 0.9930
TRAINING Epoch 37/45 Loss 0.0125 Accuracy 0.9947
TRAINING Epoch 38/45 Loss 0.0117 Accuracy 0.9947
TRAINING Epoch 39/45 Loss 0.0111 Accuracy 0.9960
TRAINING Epoch 40/45 Loss 0.0105 Accuracy 0.9967
TRAINING Epoch 41/45 Loss 0.0096 Accuracy 0.9977
TRAINING Epoch 42/45 Loss 0.0092 Accuracy 0.9970
TRAINING Epoch 43/45 Loss 0.0086 Accuracy 0.9967
TRAINING Epoch 44/45 Loss 0.0082 Accuracy 0.9963
TRAINING Epoch 45/45 Loss 0.0077 Accuracy 0.9980
Finished Training
----------
```

Figure 9

Conclusions for Part2

Fine-tuning the entire network demonstrated its benefits in terms of training accuracy. However, overfitting remains a challenge. The significant difference between training and testing accuracies suggests further optimization might be needed to improve generalization. Furthermore, using ResNet as a fixed feature extractor provided a more balanced performance between training and testing, although the overall accuracies were slightly lower.

Additional data augmentation, such as random rotation, had a negative impact on the performance of the fixed feature extractor model. This suggests that the model was sensitive to such transformations and could not generalize well with them.

For future work, exploring a balance between regularization, model complexity, and data augmentation can further optimize and balance performance between training and testing outcomes.