Lab-3-Image-Classification

ELEC 475 Prof. Michael Greenspan

Monday, November 6th, 2023

Mile Stosic (20233349)

Kieran Cosgrove (20226841)

Table of Contents

Table of Figures

1 Vanilla Architecture

In the quest for an efficient image classification model, the architecture plays an important role in determining the overall performance and accuracy. This section delves into the rationale behind the construction of our vanilla model for classifying images from the CIFAR100 dataset using fine labels.

1.1 Description

The Vanilla Model v1 represents a straightforward convolutional neural network (CNN) architecture that integrates a pre-defined encoder and a custom frontend. The encoder part is borrowed from the VGG-16 architecture, known for its simplicity and effectiveness in feature extraction from images. The encoder consists of multiple convolutional layers followed by ReLU activations and max-pooling operations that progressively down-sample the input image while increasing the depth to capture a rich set of features.

The frontend is purposely designed to be minimalistic, consisting of a flattening layer followed by a fully connected layer that maps the extracted features to the number of classes in the CIFAR-100 dataset. The motivation behind such a design was to evaluate the raw classification power of the features extracted by the VGG-16 encoder without any additional complexity. This setup ensures that the performance of the model can be attributed largely to the encoder's feature extraction capabilities.

The image accompanying this description represents a neural network diagram that illustrates the flow from input to output. Each node could represent a neuron, and the connections between them represent the neural connections that are typically weighted in a real neural network. The depth and layering pattern of the network signify the architecture's complexity and how each layer transforms the data.

In summary, the design of the Vanilla Model v1 is motivated by the goal to create a baseline for performance on the CIFAR-100 dataset, ensuring that any complexity in the model does not overshadow the fundamental feature extraction and classification process.

1.2 Diagram

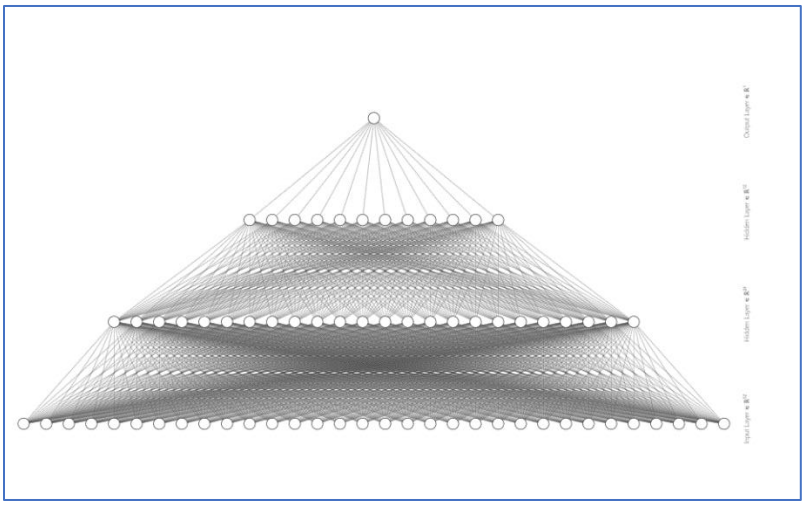

Figure 1: Example diagram for Model V1

2 Model Architecture

This Model represents an evolution of the Vanilla Model v1. The significant change introduced in this architecture is the addition of residual blocks, hence drawing inspiration from the ResNet architecture known for its ability to train much deeper networks than was previously possible.

2.1 Residual Blocks

In the new model, termed VanillaModelv2, we introduce ResidualBlock components. Each ResidualBlock consists of two convolutional layers with Batch Normalization, interspersed with ReLU activation functions. The key feature of a ResidualBlock is the addition of a shortcut connection that bypasses the two convolutional layers. This shortcut allows the gradient to flow directly through the network, mitigating the vanishing gradient problem that often occurs in deeper networks. The use of residual blocks is intended to enable the model to learn identity functions where appropriate, which can be crucial for preserving information across layers.

2.2 Backend Modification

The ResNetBackend includes the original VGG-like encoder, now followed by two of these residual blocks. The expectation is that these blocks will refine the feature maps produced by the encoder, allowing for higher-level feature representations without the degradation of training accuracy.

2.3 Frontend Compatibility

The Frontendv2 remains unchanged in its structure, still utilizing a flattening layer followed by a fully connected layer. However, we ensure the frontend is compatible with the ResNetBackend. The number of input features to the fully connected layer must match the flattened output of the ResNetBackend.

2.4 Motivation for the Model

The motivation behind incorporating residual blocks is to capture the potential for deeper feature representation without compromising the network's training efficiency. The residual blocks help address the degradation problem, where adding more layers to a deep neural network leads to higher training error, by enabling the network to skip certain layers through the use of identity shortcuts. This modification is anticipated to improve the accuracy and learning speed of the model on the complex CIFAR-100 dataset.

2.5 Diagram

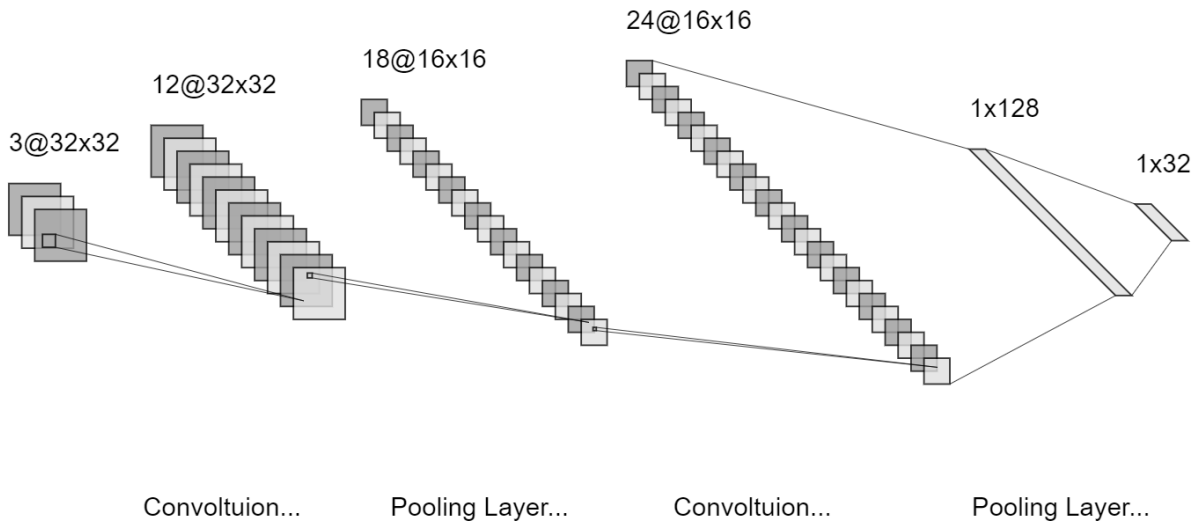

Figure 2: Example CNN diagram for model V2

3 Experiments

This section provides a comprehensive description of the training and testing procedures of our models on the CIFAR-100 dataset. Detailed here are all the steps and configurations necessary to replicate the experiments, including the hardware specifications used during the training.

3.1 Data Preparation

The CIFAR-100 dataset was used for both training and testing our models. For training, we applied data augmentation techniques such as random horizontal flipping and random cropping with padding. For both training and testing, we normalized the images using the mean and standard deviation specific to CIFAR-100.

3.2 Model Initialization

Prior to training, we initialized two architectures—Vanilla Model v1 and Vanilla Model v2—designed to compare the impact of different backend architectures on model performance. Vanilla Model v1 employed a VGG-like architecture, whereas Vanilla Model v2 was built upon a ResNet-like structure. Both models were designed to classify images into 100 distinct categories provided by the CIFAR-100 dataset.

3.3 Training Loop

The core of the training process was encapsulated within a training loop that iterated through the dataset for a pre-defined number of epochs. At each epoch, the model parameters were updated using the Stochastic Gradient Descent (SGD) optimizer. A learning rate of 0.001 and a momentum of 0.9 were chosen to guide the updates during backpropagation.

We utilized a CrossEntropyLoss function, which is suitable for multi-class classification problems like CIFAR-100. The loss was calculated for each batch, and the gradients were propagated back through the network to update the weights, with the aim of reducing the loss in subsequent iterations.

Throughout training, the running loss was tracked and averaged over the number of batches to provide insight into the model's learning progression. This metric was crucial for monitoring the training process and ensuring the model was learning effectively.

3.4 Hyperparameter Tuning

The chosen hyperparameters, such as batch size and initial learning rate, were the result of preliminary experiments designed to find a balance between training speed and convergence stability. We also experimented with different learning rate schedules and regularization techniques.

The models were trained using Stochastic Gradient Descent (SGD) with an initial learning rate of 0.001 and momentum of 0.9. The batch size was set to 128 for training and 100 for testing, and the models were trained for 10 epochs.

3.5 Model Saving and Logging

At the end of each epoch, the model's state was saved to enable the possibility of resuming training at a later time or to perform further analysis. Additionally, we maintained logs of the loss and accuracy metrics for each epoch, which facilitated the creation of the training plots included in this report.

3.6 VanillaModelv1 Testing

Analyzing the provided training output log and the corresponding plot of training loss over epochs, we can draw several conclusions about the machine learning model's performance and the nature of the training process.

The training output log shows the ten epochs of training, with the loss decreasing consistently from an initial value of approximately 4.375 to roughly 2.122 by the tenth epoch. This steady decline in loss indicates that the model is effectively learning from the training data and improving its prediction accuracy as training progresses.

Upon examining the plot of training loss, the smooth, downward trajectory of the loss curve reflects this improvement and demonstrates that the interpolation of the loss values using scipy's make_interp_spline function has created a smooth curve without any erratic fluctuations. The absence of any significant spikes or variations in the plot suggests that the training process was stable and did not encounter issues such as overfitting or difficulties with the learning rate.

Despite these positive signs of learning, the model's accuracy on the test dataset, as reported in the log, suggests that there is substantial room for improvement. A top-1 accuracy of 45.89% implies that the model's most confident predictions are correct less than half the time, while the top-5 accuracy of 75.27% indicates that the correct answer lies within the model's top five predictions approximately three-quarters of the time. This disparity between top-1 and top-5 accuracy may point to a model that is reasonably good at narrowing down the possibilities but struggles to pinpoint the precise class.

To further improve the model's accuracy, several strategies could be employed. These might include augmenting the training dataset, implementing regularization techniques to combat overfitting if it is

occurring, tuning hyperparameters like the learning rate, experimenting with different model architectures, or simply continuing the training for more epochs, especially if the loss continues to decline. Each of these steps could potentially lead to better performance, especially when applied judiciously based on a more in-depth analysis of the training and validation dynamics.

```
Files already downloaded and verified
Files already downloaded and verified
Epoch 1, Loss: 4.3752356356062245
Epoch 2, Loss: 3.8136809994192684
Epoch 3, Loss: 3.487574805993863
Epoch 4, Loss: 3.240595585854767
Epoch 5, Loss: 2.9954714732401815
Epoch 6, Loss: 2.8072630974947645
Epoch 7, Loss: 2.599314179261932
Epoch 8, Loss: 2.430079522340194
Epoch 9, Loss: 2.2747868308630745
Epoch 10, Loss: 2.1223129738322304
Finished Training
Accuracy of the network on the 10000 test images:
Top 1 accuracy: 45.89%
Top 5 accuracy: 75.27%
```

Figure 3: Test results from V1

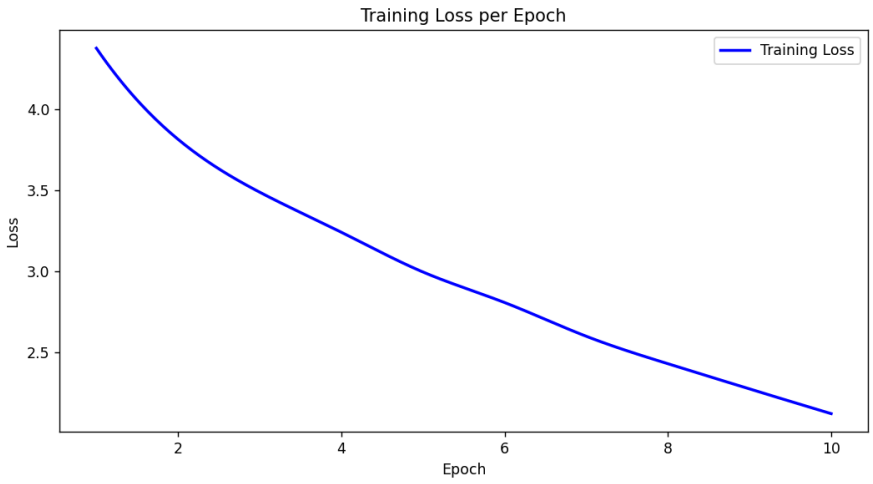

Figure 4: Loss Curve for V1

3.7 VanillaModelv2 Testing

The additional information provided by the updated second model gives a clearer picture of the machine learning model's training progression and performance on the test dataset.

From the log, we observe a consistent decline in training loss over the span of 10 epochs, starting at 3.9397 in epoch 1 and steadily decreasing to 1.7301 by epoch 10. This suggests that the model is learning and improving its ability to fit the training data over time. Such a consistent decline is indicative of a well-tuned learning rate where the model is neither learning too slowly nor making updates that are too large, which can lead to erratic loss trajectories.

The smooth curve in the loss plot confirms this consistent decline in training loss, visualizing the model's improvement across epochs without any noticeable anomalies such as increases in loss which could suggest problems like overfitting or catastrophic forgetting. The absence of such issues in the loss plot is a good sign that the model's learning process is stable.

Looking at the test accuracy, the top-1 accuracy is reported as 48.18%, and the top-5 accuracy as 77.42%. A top-1 accuracy of slightly over 48% indicates that the model correctly predicts the exact class less than half the time, which, depending on the complexity of the task and the dataset, might be an acceptable starting point. The relatively high top-5 accuracy suggests the model has a better grasp of the possible classes and could benefit from refinement in distinguishing among the top probable categories.

```
Files already downloaded and verified
Files already downloaded and verified
Epoch 1, Loss: 3.9396966028091547
Epoch 2, Loss: 3.3049474860091346
Epoch 3, Loss: 2.9784592838238573
Epoch 4, Loss: 2.7035970986651643
Epoch 5, Loss: 2.446219739401737
Epoch 6, Loss: 2.2478827863093227
Epoch 7, Loss: 2.0905780453816094
Epoch 8, Loss: 1.9526998975392802
Epoch 9, Loss: 1.8419174846175992
Epoch 10, Loss: 1.7301099336970494
Finished Training
Accuracy of the network on the 10000 test images:
Top 1 accuracy: 48.18%
Top 5 accuracy: 77.42%
```

Figure 5: Test results from V2

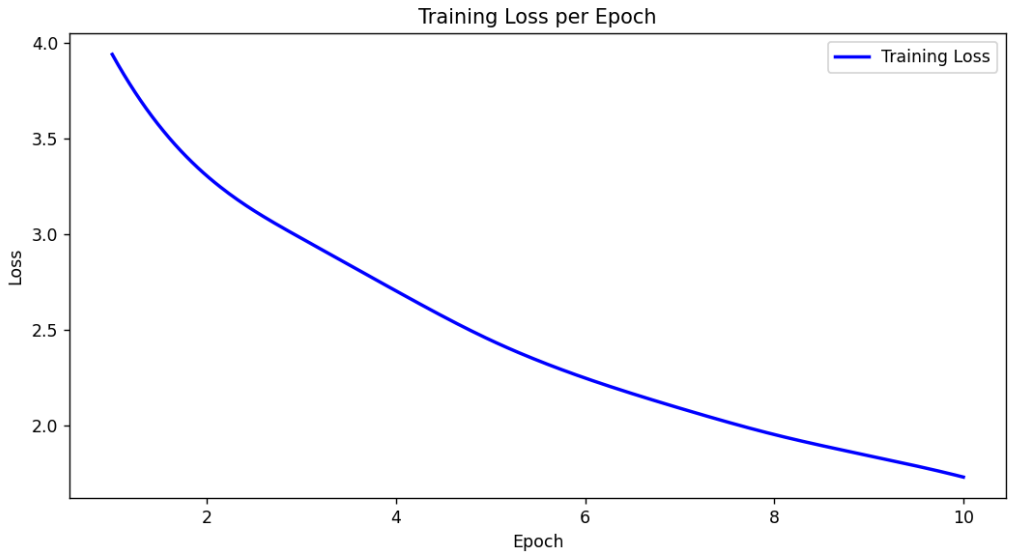

Figure 6: Loss Curve from V2

3.8 Resource Management (Hardware)

The experiments were performed on a dedicated GPU (NVIDIA GeForce GTX 1650) to leverage parallel computation capabilities, thus speeding up the training process. The choice of the device was made dynamically based on the availability of CUDA-compatible hardware. If CUDA was unavailable, the model defaulted to CPU training, which, while slower, ensured that the training could still be completed.

4 Conclusion

The comparative analysis of the two neural network models on the CIFAR-100 dataset highlighted distinct performance metrics. The simple CNN (Vanilla Model v1) achieved a top-1 accuracy of 45.89%, setting a solid baseline. On the other hand, the enhanced CNN with ResNet-style residual blocks (Vanilla Model v2) outperformed the baseline with a top-1 accuracy of 48.18%, affirming the benefits of deeper architectures and residual connections for complex image classification tasks.

Despite the complexity and increased resource demands of the modified model, the performance gains justify its use for the CIFAR-100 dataset. The conclusion drawn from these experiments suggests that strategic architectural enhancements can lead to better model performance, though careful consideration must be given to balance accuracy, computational cost, and model complexity. Future efforts could further optimize this balance for improved outcomes.