REPORT - MINOR 02

PRINCY GAUTAM
B20BB051

Question 01

The given code implements a knowledge distillation technique, where a complex teacher model is trained and its knowledge is transferred to a simpler student model, to achieve better performance than the student model trained alone. The dataset used for this task is SVHN (Street View House Numbers), which consists of images of house numbers from Google Street View, which are cropped and centered to be 32x32 pixels in size. The task is to classify these images into 10 classes (digits 0-9).

Data Preprocessing

The data preprocessing is performed using PyTorch's `torchvision.transforms` module. The transformation includes resizing the images to 32x32, converting them to tensors, and normalizing the pixel values with mean and standard deviation of 0.5.

Model Architecture

Teacher Network

The teacher network is a deep convolutional neural network consisting of 8 convolutional layers and 2 fully connected layers. Each convolutional layer is followed by a ReLU activation function and a 2x2 average pooling layer. The output of the last convolutional layer is flattened and passed through two fully connected layers with ReLU activation in between. The output layer has 10 units, representing the 10 classes of SVHN dataset.

Student Network

The student network is a simpler version of the teacher network, consisting of only 2 convolutional layers and 2 fully connected layers. Each convolutional layer is followed by a ReLU activation function and a 2x2 average pooling layer. The output of the last convolutional layer is flattened and

passed through a fully connected layer with ReLU activation. The output layer has 10 units, representing the 10 classes of SVHN dataset.

Training and Optimization

The teacher network is first trained on the SVHN dataset using the Adam optimizer and the cross-entropy loss function. The student network is initialized and trained using the same optimizer and loss function. The `xavier_weight()` function is used to initialize the weights of both networks with Xavier initialization, which helps in improving the convergence speed of the network.

Observation

Loss measured after training teacher network is as follows:

```
Train Teacher Network: Epoch 0     Loss: 0.694274
Train Teacher Network: Epoch 1     Loss: 0.353693
Train Teacher Network: Epoch 2     Loss: 0.264385
Train Teacher Network: Epoch 3     Loss: 0.217452
Train Teacher Network: Epoch 4     Loss: 0.185256
Train Teacher Network: Epoch 5     Loss: 0.155838
Train Teacher Network: Epoch 6     Loss: 0.129749
Train Teacher Network: Epoch 7     Loss: 0.106129
Train Teacher Network: Epoch 8     Loss: 0.082741
Train Teacher Network: Epoch 9     Loss: 0.065557
```

Loss measured after training student network with knowledge distillation is as follows:

```
Student Network: Epoch 1     Loss: 1.180264
Student Network: Test Loss: 0.000017, Accuracy: 84.79%
Student Network: Epoch 2     Loss: 1.051913
Student Network: Test Loss: 0.000015, Accuracy: 86.06%
Student Network: Epoch 3     Loss: 0.939397
Student Network: Test Loss: 0.000015, Accuracy: 86.33%
Student Network: Epoch 4     Loss: 0.847877
Student Network: Test Loss: 0.000013, Accuracy: 87.19%
Student Network: Epoch 5     Loss: 0.771190
Student Network: Test Loss: 0.000014, Accuracy: 87.14%
Student Network: Epoch 6     Loss: 0.702909
Student Network: Test Loss: 0.000016, Accuracy: 87.48%
Student Network: Epoch 7     Loss: 0.649012
```

```
Student Network: Test Loss: 0.000014, Accuracy: 87.55%
Student Network: Epoch 8     Loss: 0.604804
Student Network: Test Loss: 0.000013, Accuracy: 88.16%
Student Network: Epoch 9     Loss: 0.564236
Student Network: Test Loss: 0.000013, Accuracy: 88.46%
Student Network: Epoch 10    Loss: 0.530408
Student Network: Test Loss: 0.000013, Accuracy: 88.28%
Student Network: Epoch 11    Loss: 0.496937
Student Network: Test Loss: 0.000014, Accuracy: 88.46%
Student Network: Epoch 12    Loss: 0.463040
Student Network: Test Loss: 0.000013, Accuracy: 88.29%
Student Network: Epoch 13    Loss: 0.441367
Student Network: Test Loss: 0.000014, Accuracy: 88.50%
Student Network: Epoch 14    Loss: 0.413029
Student Network: Test Loss: 0.000015, Accuracy: 87.74%
```

Results

The student network trained using the knowledge distillation technique performs better than the same student network trained without the teacher network. This is because the student network is trained to mimic the output of the teacher network, which is expected to have better performance due to its complexity and deeper architecture.

Conclusions

The given code implements the knowledge distillation technique to transfer the knowledge of a complex teacher network to a simpler student network. The implementation uses PyTorch's built-in functionality to preprocess the data, define the model architectures, initialize the weights, and train the networks.

Question 02

Q1. How can we use transfer learning to improve the performance of an image classification model on a new dataset with limited labeled data?

Answer : To use transfer learning for improving the performance of an image classification model on a new dataset with limited labeled data, we can take the following steps:

1. Choose a pre-trained model: Select a pre-trained model that has been trained on a large dataset, such as VGG, ResNet, or Inception, that has achieved good performance on a similar task as the one you want to solve.
2. Remove the classification layer: Since the pre-trained model is trained on a different dataset and has a different number of output classes, we need to remove the classification layer and replace it with a new layer with the number of output classes we need for our task.
3. Freeze the pre-trained layers: To prevent overfitting and ensure that the pre-trained weights are not significantly altered, we can freeze the weights of the pre-trained layers, which means they will not be updated during training.
4. Train the model: Train the modified model on the new dataset with limited labeled data, using the frozen pre-trained layers as feature extractors, and only updating the weights of the new classification layer. Fine-tune the model by gradually unfreezing the pre-trained layers and training the entire model on the new dataset.

By using transfer learning in this way, we can leverage the knowledge learned by the pre-trained model on a large dataset to improve the performance of the model on a new dataset with limited labeled data.

Justification

This approach is widely used in the deep learning community and has been shown to be effective in many practical applications.

1. Selecting a pre-trained model that has achieved good performance on a similar task as the one we want to solve is a reasonable step.

2. Removing the classification layer and replacing it with a new layer with the number of output classes we need is necessary to ensure that the model can be applied to our specific task.

3. Freezing the pre-trained layers is also an important step because it ensures that the weights of the pre-trained layers are not significantly altered during training, which can prevent overfitting and help ensure that the learned features are still relevant.

Q2. How can we leverage the interpretability of deep learning algorithms to improve the transparency and trustworthiness of image classification systems used in industries such as healthcare and finance?

Answer : One approach to increasing interpretability is to use visualization techniques such as saliency maps and activation maximization to highlight which regions of an input image the model is focusing on to make its classification decision. This can help us understand which features are most important for classification, and provide insights into how the model is making its decisions.

Another approach is to use techniques such as model distillation or pruning to simplify the model architecture and reduce its complexity. This can make the model easier to understand and interpret, while still maintaining a high level of accuracy.

In addition, techniques such as LIME (Local Interpretable Model-Agnostic Explanations) and SHAP (Shapley Additive Explanations) can provide explanations of individual predictions, by approximating the model locally around the input and computing feature attributions that help explain why the model made a particular decision.
Finally, using transfer learning and fine-tuning pre-trained models can provide insights into the learned representations and features used for classification. By examining the pre-trained model's activations and weights, we can gain insights into what features and patterns the model has learned to recognize, and how these features contribute to the model's overall performance.
Overall, by leveraging interpretability techniques, we can improve the transparency and trustworthiness of deep learning-based image classification systems, and make them more accessible and understandable to end-users and stakeholders in various industries.

Justification

The answer provided outlines several ways in which the interpretability of deep learning algorithms can be leveraged to improve the transparency and trustworthiness of image classification systems in industries such as healthcare and finance.
The use of visualization techniques such as saliency maps and activation maximization can help us understand which features are important for classification and how the model is making its decisions. This approach can be particularly useful in industries where the stakes are high, such as healthcare and finance, as it can help us identify the factors that are driving the model's predictions.
Simplifying the model architecture using techniques such as model distillation or pruning can also improve interpretability by reducing complexity and making it easier to understand the factors that are contributing to the model's predictions.

Q3. Given a time series dataset of daily stock prices for a company, use an RNN to predict the stock prices for the next 5 days. The dataset consists of 500 daily closing prices.
(a) Describe how you would preprocess the dataset for training an RNN.
(b) Train an RNN model using the preprocessed dataset and evaluate its performance on a validation set.
(c) Suppose the model achieves a mean absolute error of 2.5 dollars on the validation set. Describe how you could modify the model architecture or hyperparameters to improve its performance.

Answer: In general some possible ways to improve the performance of an RNN model might include adjusting the number of layers or hidden units in the model, changing the activation function or regularization techniques used, adjusting the learning rate or optimizer used for training, or incorporating additional features or data sources into the model.
It's important to note that improving the performance of an RNN model can be a challenging and iterative process, and may require a combination of experimentation, analysis, and domain expertise.

Justification

The above answer correctly notes that the specific modifications needed to improve the performance of an RNN model would depend on a variety of factors, including the architecture of the model and the characteristics of the dataset being used.

In general, there are many possible ways to modify an RNN model to improve its performance, such as increasing the number of layers or hidden units, changing the activation function, adding regularization techniques like dropout, and adjusting the learning rate or optimizer used for training. However, the effectiveness of these modifications can depend on many factors, such as the size and complexity of the dataset, the presence of noisy or missing data, and the specific learning task being performed.

Furthermore, improving the performance of an RNN model is often an iterative process that requires careful experimentation and analysis to identify the most effective modifications. This process may require a combination of domain expertise, statistical analysis, and trial-and-error experimentation to find the best possible model configuration for the given task and dataset. Therefore, the answer provided is accurate in emphasizing the importance of careful experimentation and analysis when attempting to improve the performance of an RNN model.