DEEP LEARNING AND COMPUTER VISION APPLICATIONS: PROJECT ON TRANSFER LEARNING

Under the guidance of Prof. Dr. Shitala Prasad

APRIL 1, 2023

SUBMITTED BY:
Umang Srivastav (2213108); Srivishnu TN

Table of Contents

Acknowledgement

I would like to express my sincere gratitude to my professor Prof. Dr. Shitala Prasad for their guidance and support throughout the course of this deep learning project. His insights and feedback were invaluable in helping me to develop a deep understanding of the subject matter and to produce a high-quality project report.

I would like to thank all the authors and researchers whose work I have referred to in this report. Their contribution to the field of deep learning has been instrumental in helping me to gain a deeper understanding of the subject matter.

Thank you all for your support and encouragement throughout this project.

Sincerely,

Umang Srivastav

Abstract

Convolutional Neural Networks (CNNs) have shown remarkable performance in image classification tasks. In this paper, we investigate the transfer learning approach for improving the performance of ResNet-50 on CIFAR-100 dataset. Specifically, we first train ResNet-50 on CIFAR-10 dataset until it converges and then shrink the layers to 18 and fine-tune the network for CIFAR-100 dataset. We compare the performance of this strategy with a purely 18-layered ResNet trained on CIFAR-100 for the same number of epochs. Our experimental results demonstrate that the transfer learning approach significantly outperforms the purely 18-layered ResNet in terms of accuracy and convergence speed.

Moreover, we propose a novel way of training the network where the original data is seen only for a few-shot. We use the pre-trained ResNet-50 on CIFAR-10 dataset as a feature extractor and train a few-shot learning model using Prototypical Networks. The proposed approach achieves competitive accuracy with fewer training samples than the traditional training approach.

Overall, our study provides insights into the transfer learning approach and few-shot learning for improving the performance of ResNet on CIFAR-100 dataset. The proposed methods can be further applied to other image classification tasks with limited training data.

Resnet 50

Layer Type	Output Shape	Parameters
Input	(224, 224, 3)	0
Convolution	(112, 112, 64)	9,408
Max Pooling	(56, 56, 64)	0
Residual Block	(56, 56, 256)	75,008
Residual Block	(56, 56, 256)	100,353
Residual Block	(56, 56, 256)	100,353
Residual Block	(28, 28, 512)	379,392
Residual Block	(28, 28, 512)	520,192
Residual Block	(28, 28, 512)	520,192
Residual Block	(28, 28, 512)	520,192
Residual Block	(14, 14, 1024)	3,675,392
Residual Block	(14, 14, 1024)	2,097,152
Residual Block	(14, 14, 1024)	2,097,152
Residual Block	(14, 14, 1024)	2,097,152
Residual Block	(14, 14, 1024)	2,097,152
Residual Block	(7, 7, 2048)	14,745,088
Residual Block	(7, 7, 2048)	8,388,608
Residual Block	(7, 7, 2048)	8,388,608
Average Pooling	(1, 1, 2048)	0
Fully Connected	(1, 1, 1000)	2,049,000
Output	(1, 1, 1000)	0

ResNet-50 is a deep neural network architecture with 50 layers that was introduced by Microsoft Research Asia. The table above describes the different layers in the ResNet-50 architecture along with their output shapes and corresponding parameter counts. The input to ResNet-50 is a 224x224x3 image, and the final output is a probability distribution over 1000 classes.

The ResNet-50 architecture is composed of residual blocks, which allow for the training of much deeper neural networks without the problem of vanishing gradients. Each residual block contains several convolutional layers, followed by batch normalization and activation functions, and then a skip connection that adds the input to the output of the convolutional layers. The architecture also uses average pooling and fully connected layers to produce the final output.

Resnet 18

Layer Type	Output Shape	Parameters
Input	(224, 224, 3)	0
Convolution	(112, 112, 64)	1,792
Max Pooling	(56, 56, 64)	0
Residual Block	(56, 56, 64)	74,752
Residual Block	(28, 28, 128)	219,904
Residual Block	(14, 14, 256)	879,616
Residual Block	(7, 7, 512)	3,487,232
Average Pooling	(1, 1, 512)	0
Fully Connected	(1, 1, 1000)	513,000
Output	(1, 1, 1000)	0

ResNet-18 is a variant of the ResNet architecture that has 18 layers. It was also introduced by Microsoft Research Asia and is commonly used for image classification tasks. The architecture of ResNet-18 is similar to ResNet-50, but it is much smaller and shallower.

The input to ResNet-18 is also a 224x224x3 image, and the final output is a probability distribution over 1000 classes. The first layer of ResNet-18 is a convolutional layer with 64 filters, followed by a max pooling layer. The architecture then contains four residual blocks, each containing two convolutional layers, batch normalization, and an activation function. The residual blocks also include skip connections to preserve information from earlier layers.

The number of filters in each residual block increases as the spatial resolution decreases. Specifically, the first residual block has 64 filters, the second has 128 filters, the third has 256 filters, and the fourth has 512 filters. After the last residual block, the architecture uses average pooling and fully connected layers to produce the final output.

Overall, ResNet-18 is a smaller and more computationally efficient variant of ResNet that still achieves competitive performance on many image classification tasks. It is often used as a baseline architecture for comparison with more complex models.

Transfer Learning

Transfer learning is a machine learning technique that allows the knowledge gained while solving one problem to be applied to a different but related problem. In the context of deep learning, transfer learning is often used to speed up the training of deep neural networks by reusing the pre-trained weights from a model that has already been trained on a large dataset.

The basic idea of transfer learning is to use the pre-trained weights from a model as a starting point for a new model, rather than initializing the weights randomly. This can be especially useful when working with limited amounts of data, as it allows the new model to benefit from the knowledge gained by the pre-trained model.

There are two main types of transfer learning: fine-tuning and feature extraction. Fine-tuning involves taking a pre-trained model and continuing the training on a new dataset, allowing the weights to be adjusted to better fit the new data. Feature extraction involves using the pre-trained model as a fixed feature extractor, removing the last few layers of the model and replacing them with new layers that are trained on the new dataset.

One of the most common uses of transfer learning in deep learning is in the field of computer vision. For example, pre-trained models such as VGG, ResNet, and Inception have been trained on large image datasets such as ImageNet, and their pre-trained weights can be used as a starting point for training new models on smaller image datasets.

Transfer learning has also been successfully applied in other areas of deep learning, such as natural language processing and speech recognition. In these domains, pre-trained models such as BERT and GPT-2 have been used as a starting point for new models that are trained on smaller datasets.

Overall, transfer learning is a powerful technique that can help improve the performance of deep learning models, especially when working with limited amounts of data. By reusing pre-trained weights from a model that has already learned a lot about a related task, we can speed up the training process and improve the accuracy of our models.

Few Shots Learning

Few-shot learning is a subfield of machine learning that focuses on training models to recognize new classes of objects or concepts with only a few examples. Traditional machine learning algorithms often require large amounts of data to achieve good performance, but few-shot learning aims to generalize from just a few examples by leveraging prior knowledge and transfer learning.

The basic idea of few-shot learning is to train a model on a few examples of each class, and then use that model to recognize new examples of those classes. This can be challenging because the model must be able to generalize from a few examples to a whole class, and must be able to differentiate between new classes that are similar to each other.

One approach to few-shot learning is to use meta-learning, which involves training a model to learn how to learn. Specifically, the model is trained on a large number of small "meta-tasks," each of which consists of a few examples of a new class. The model is then tested on its ability to recognize new classes that it has not seen before. By training the model on a large number of meta-tasks, it learns to quickly adapt to new classes and generalize from a few examples.

Another approach to few-shot learning is to use generative models, which can generate new examples of a class given a few examples as input. For example, a generative model such as a variational autoencoder or a generative adversarial network can be trained on a few examples of a class, and then used to generate new examples of that class that can be used to train a classifier.

Few-shot learning has many potential applications, such as in robotics, where robots must quickly adapt to new tasks and environments, and in healthcare, where models must be able to recognize new diseases and symptoms with only a few examples. However, few-shot learning is still a relatively new field, and there are many challenges that must be addressed, such as how to ensure that the model is able to generalize to new classes and how to evaluate the performance of few-shot learning models.

Datasets

CIFAR-10 The CIFAR-10 dataset is a collection of 60,000 32x32 color images in 10 classes, with 6,000 images per class. The classes are:

- airplane
- automobile
- bird
- cat
- deer
- dog
- frog
- horse
- ship
- truck

The dataset is split into 50,000 training images and 10,000 test images. The images are of low resolution and are relatively easy to classify, making them a good benchmark dataset for computer vision tasks.

CIFAR-10 has been used as a benchmark dataset for a wide range of computer vision tasks, including image classification, object detection, and semantic segmentation. Many state-of-the-art models in computer vision have been trained on the CIFAR-10 dataset.

CIFAR-100 The CIFAR-100 dataset is similar to CIFAR-10, but contains 100 classes, with 600 images per class. The classes are grouped into 20 super classes, each containing 5 subclasses. The subclasses are more fine-grained than the classes in CIFAR-10 and include things like fruits, vegetables, and insects.

The dataset is split into 50,000 training images and 10,000 test images. Like CIFAR-10, the images are of low resolution and are relatively easy to classify, but the increased number of classes and subclasses makes the task more challenging.

CIFAR-100 is often used as a benchmark dataset for fine-grained image classification, where the goal is to recognize subtle differences between similar objects. Many state-of-the-art models in fine-grained image classification have been trained on the CIFAR-100 dataset.

Execution

1. Python: Make sure you have Python installed on your system. PyTorch requires Python 3.6 or higher.

2. PyTorch: Install PyTorch library which provides various modules and functions for deep learning. You can install it using pip or conda package manager.

3. Jupyter Notebook: Install Jupyter Notebook which is an open-source web application that allows you to create and share documents that contain live code, equations, visualizations, and narrative text. Google Colab or Kaggle are other options to use.

4. Dataset: Datasets are installed from PyTorch library torchvision.datasets. SO external files are not required to be loaded for data for the same.

5. Hardware requirements: As the size of the datasets are large, you may need a system with a decent CPU and GPU to train your models efficiently. You can use cloud-based services such as Google Colab, AWS, or Azure if you don't have access to a high-performance machine.

Once you have installed all the necessary libraries and set up your environment, you can execute your Jupyter Notebook files on PyTorch for ResNet-18 and ResNet-50 transfer learning.

Observations

RESNET-50 on CIFAR-10 has an accuracy (test) as high as 80%.

Now, a **modified model**, the first 18 layers are extracted from the above trained model as:

```
        resnet=model
        self.conv1=resnet.conv1
        self.bn1 = resnet.bn1
        self.relu = resnet.relu
        self.maxpool = resnet.maxpool
        self.layer1 = resnet.layer1
        self.layer2 = resnet.layer2
        del self.layer2[3]
        self.avgpool = resnet.avgpool
        self.fc=torch.nn.Linear(in_features=512, out_features=100, bias
=True)
```

Now, another model gets made using those 18 layers and a fully connected layer getting attached to it at last:

```
        x = self.conv1(x)
        x = self.bn1(x)
        x = self.relu(x)
        x = self.maxpool(x)
        x = self.layer1(x)
        x = self.layer2(x)
        x = self.avgpool(x)
        x = torch.flatten(x, 1)
        x = self.fc(x)
```

Then all the layers' parameters, except the last FC layer, are kept fixed (not trainable):

```
for name, param in model.named_parameters():
    if name.startswith('fc'):
        param.requires_grad = True
    else:
        param.requires_grad = False
```

And then the whole model is trained again on CIFAR-100. This is Transfer learning from one model to another.
The accuracy of this model on **CIFAR-100** turns out to be around 40%.

But when the same **modified model** is fine tuned on **CIFAR-10**, the performance turned out to be almost equal as the original RESNET-50 trained on CIFAR-10.
The first 18 layers of ResNet50 learned useful features for CIFAR10, which were then used to improve the performance of a new model on that same dataset.

The **RESNET-18** trained on **CIFAR-100** gives the test accuracy as high as 48%.

Some extra work: Few Shots Learning

Three different ways of few shots learning are done here:

1. Using just 10% of the dataset to train the model of Resnet50 on CIFAR10, the test accuracy turned out to be 57%.

2. Another method used here is data augmentation with transfer learning (pretrained Resnet50 on imagenet), but there was no significant impact on test accuracy as it turned out to be just 10-12%.

3. The third method is using just few samples (50) of each class and then train the model (Resnet50 on CIFAR10) with those sample and then put it to test. The accuracy turned out to be just 12%.

More work needs to be done in this area as this was just some trial. The first method turned out to be favourable to the learning. But, the other two methods also can be fine tuned more.

Conclusions

1. Transfer learning using pre-trained models can be effective: By using the pre-trained ResNet50 model on CIFAR10 and modifying it for CIFAR100, you were able to achieve a 40% accuracy rate. This shows that transfer learning can be a powerful technique for improving model performance on new datasets.

2. The performance of the modified model depends on the similarity between the source and target datasets: While the ResNet50 model was originally trained on CIFAR10, it was still able to provide some benefits for CIFAR100. However, the accuracy rate of 40% suggests that there may be significant differences between the two datasets. This highlights the importance of choosing a pre-trained model that is as similar as possible to the target dataset.

3. Model architecture plays an important role in performance: The fact that a ResNet18 model trained from scratch on CIFAR100 achieved a higher accuracy rate of 48% compared to the modified ResNet50 model suggests that the ResNet18 architecture may be better suited to the task. It is possible that the additional layers in the ResNet50 model, which were designed for more complex datasets, may not have been as helpful for CIFAR100.

4. As the modified model with the first 18 layers of ResNet50 fixed and a trainable fully connected layer at the end achieved nearly the same performance as the original ResNet50 model on CIFAR10, it suggests that the first 18 layers of ResNet50 are already well-suited for the CIFAR10 dataset. In other words, these layers have learned to extract features that are relevant for the CIFAR10 dataset, and this knowledge can be transferred to a new model without the need for further fine-tuning.

5. This result is consistent with the idea of transfer learning, which involves leveraging knowledge learned from a source dataset to improve performance on a target dataset. In this case, the first 18 layers of ResNet50 learned useful features for CIFAR10, which were then used to improve the performance of a new model on that same dataset.

6. Overall, this result suggests that the effectiveness of transfer learning depends on the similarity between the source and target datasets, and that pre-trained models can be a useful starting point for improving model performance on related tasks.

References

1. https://journalofbigdata.springeropen.com/articles/10.1186/s40537-016-0043-6

2. https://arxiv.org/abs/1512.03385v1

3. https://arxiv.org/abs/2203.04291