Multi-Task Learning for Image Classification: A Vision Transformers Approach

Minh Nguyen
Virginia Tech
mnguyen0226@vt.edu

Peter Chen
Virginia Tech
chenp3@vt.edu

Abstract

Image classification is a fundamental task in computer vision, with many practical applications. However, achieving high accuracy on complex datasets like Cifar-10 and Cifar-100 can be challenging, particularly when working with limited computational resources. In this study, we evaluate the performance of Multi-Task Learning (MTL) on two common image classification tasks: class and superclass classification. Specifically, we investigate the efficiency of MTL using the Vision Transformers (ViT) architecture on these datasets, shedding light on its comparative performance against the Single-Task Learning (STL) approach and traditional convolution-based models. Our results demonstrate that MTL using the ViT architecture outperforms the STL approach on both classification tasks, achieving higher accuracy with fewer parameters. Additionally, we find that ViT and ResNet-152 perform similarly on these tasks, highlighting the potential of ViT for MTL scenarios. These findings have important implications for the development of efficient and effective image classification models, particularly in scenarios where multiple classification tasks need to be performed simultaneously.

The code, results, and commit contribution can be found at: `https://github.com/mnguyen0226/multitask_learning_vit`.

1. Introduction

In recent years, Multi-Task Learning (MTL) has emerged as a promising approach to tackle multiple tasks simultaneously while optimizing computational resources [3]. In computer vision, MTL has shown potential in addressing challenges such as image segmentation, keypoint detection, and edge detection, with demonstrated improvements in data efficiency and performance on related tasks [5, 3].

The Vision Transformer (ViT) is a cutting-edge neural network architecture inspired by the Transformer model used in natural language processing. ViT represents images as fixed-size patches and employs a self-attention mechanism to capture global information, allowing it to outperform convolution-based models in image classification tasks [4]. Given its ability to capture global features and locations, ViT presents a compelling framework for Multi-Task Learning.

In this study, we investigate the effectiveness of Multi-Task Learning using the Vision Transformer architecture for image classification. We focus on class and superclass classification tasks on the widely used Cifar-10 and Cifar-100 datasets, comparing the performance of MTL with the Single-Task Learning (STL) approach and traditional convolution-based models. Specifically, we aim to demonstrate that MTL using the ViT architecture outperforms the STL approach while providing similar performance to convolution-based models. Our results will offer valuable insights into the potential of ViT in Multi-Task Learning scenarios and enrich our understanding of its effectiveness in diverse image classification tasks.

2. Methods

This section details the architecture of Vision Transformers (ViT), Residual Network architecture, Multi-Task Learning, research questions, and experimental design.

2.1. Vision Transformers (ViT)

In this study, we compare the ViT architecture with the Convolutional Neural Network (CNN) architecture. ViT is an encoder-based transformer neural network that uses a self-attention mechanism to transform the input image into fixed-size patches and encode their positions into the input, allowing the network to capture global features and their locations. In contrast, CNNs focus on extracting local features. The ViT architecture consists of the following steps: input, linear projection, stacked encoder, multi-layer perceptron, and output labels. Figure 1 shows the architecture of ViT.

The explanation of ViT is cited from the original paper "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale" by Dosovitskiy et al. [4]. The image is first converted into patches, and each patch is reshaped into a 1D vector that is multiplied by a learnable matrix to

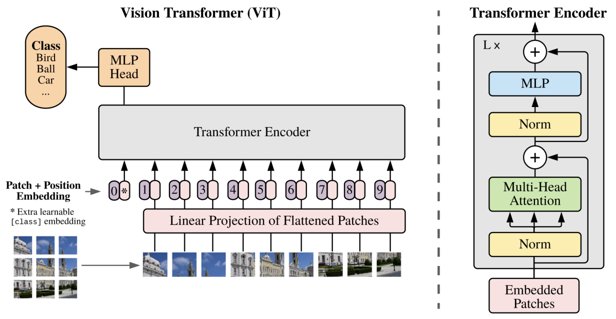


Figure 1. Vision Transformers Architecture. [4]

create a vector. Next, positional embedding is integrated as ViT is invariant to position on patches. The encoder of ViT is copied directly from the original Transformer architecture. The embedded patches are passed through a Layer Normalization to reduce training time and stabilize the training phases. Then, we pass the information through a multi-head attention network added by a skip connection to improve the performance while reducing the risk of gradient explosion or vanishing. We then pass through another Layer Normalization, Multi-layer Perceptron, and skip connection for further processing. The use of positional embedding allows ViT to behave like CNN, and this is the only inductive bias in ViT. Compared with CNN, ViT allows the network to learn the global and abstract representations of the input image, making it more robust.

2.2. Residual Network 152 (ResNet-152)

Residual Network 152, also known as ResNet-152, is a convolutional neural network architecture designed to address the vanishing gradient problem in building deep neural networks. [6] It was introduced in the original paper ”Deep residual learning for image recognition” by He et al. [6].

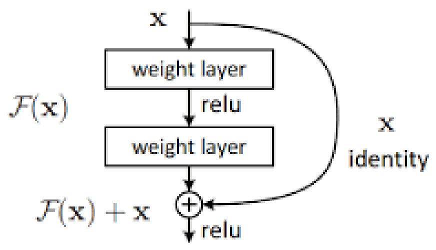

x
weight layer
$\mathcal{F}(\mathbf{x})$
relu
weight layer
x
identity
$\mathcal{F}(\mathbf{x}) + \mathbf{x}$
relu

Figure 2. Residual Connection Block Architecture. [4]

The key innovation of ResNet-152 is the residual block, which allows the network to learn residual functions that map the input to the output, rather than trying to learn the entire mapping in one shot [6]. The residual block consists of two convolutional layers with batch normalization and ReLU activation, and a skip connection that adds the input of the block to the output of the second convolutional layer, as shown in figure 2 [6].

Due to its depth, ResNet-152 can capture more complex features and achieve better performance than its shallower counterparts, such as ResNet-18, ResNet-34, and ResNet-50 [6]. It has been used in various computer vision tasks, such as object detection, image classification, and image segmentation that we learned in class.

2.3. Multi-Task Learning (MTL)

Multi-Task Learning (MTL) is a powerful technique that allows networks to learn multiple related tasks simultaneously, rather than training separate models for each task. Compared to Single-Task Learning (STL), MTL offers better efficiency and generalization, making it a popular approach in various fields [11].

Efficiency is a critical factor in embedded applications and deployment, especially due to hardware limitations and cloud storage costs. On the other hand, generalization is a crucial step towards building artificial generalized intelligence. MTL can help achieve both of these goals.

There are two main ways to implement MTL: ”hard-parameter sharing” and ”soft-parameter” sharing [10] [1] [14]. Hard-parameter sharing involves sharing some or all of the layers between tasks, while soft-parameter sharing involves learning task-specific parameters alongside shared ones.

In our project, we will be using the hard-parameter sharing approach, where the backbone of the two networks will be ViT and ResNet-152. Both branches of the MTL model will be similar in both networks, allowing for efficient training and sharing of knowledge between tasks.

2.4. Research Questions

In our project, we aim to address the following research questions:

1. **Research Question 1:** Is an MTL ViT model superior in performance to an MTL Convolution-based model (ResNet-152)?

2. **Research Question 2:** Can an MTL ViT model achieve better results than two separate STL ViT models?

3. **Research Question 3:** Does an STL ViT model outperform an STL Convolution-based model (ResNet-152) in terms of accuracy?

2.5. Experimental Design

While there are numerous benchmark datasets available for MTL, such as MS-COCO, CityScapes, and Taskonomy, our computational resources, including ARC, are limited and cannot accommodate processing such large-scale datasets [8] [2] [13]. Therefore, we have opted to use Cifar-10 and Cifar-100 as our benchmark datasets. These datasets offer a diverse range of tasks for evaluation purposes.

For the Cifar-10 dataset, we have selected two tasks: Task 1 involves a 10-class classification, while Task 2 focuses on binary classification by categorizing the 10 classes into "animal" or "vehicle" labels. For the Cifar-100 dataset, we have identified two tasks: Task 1 encompasses a 100-class classification, and Task 2 involves a 20-superclass classification, where the 100 classes are grouped into 20 superclasses, such as aquatic mammals, fish, flowers, and food containers [7]. Additional information about the superclasses can be found in the Cifar dataset documentation [7].

After data processing, we will have a total of 12 Jupyter Notebooks, each representing one of 12 different experiments with unique configurations. All experiments will be conducted from scratch. The 12 experiments will be divided as follows:

1. Train, validate, and test STL ResNet152 on Cifar-10 and Cifar-100 for each task (4 notebooks).

2. Train, validate, and test STL ViT on Cifar-10 and Cifar-100 for each task (4 notebooks).

3. Train, validate, and test MTL ResNet152 on Cifar-10 and Cifar-100 for each task (2 notebooks).

4. Train, validate, and test MTL ViT on Cifar-10 and Cifar-100 for each task (2 notebooks).

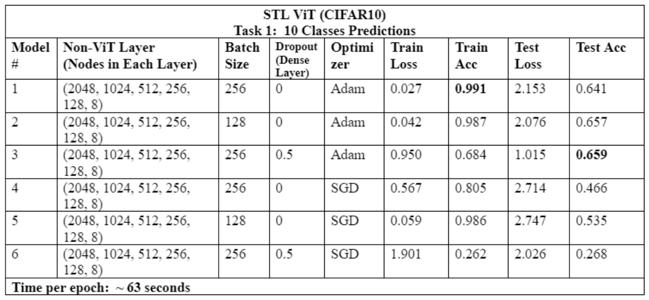
During the training and validation process, we will tune the architectures for the best performance, compare the results using classification metrics (accuracy and loss), and answer the three research questions posed in this study.

3. Experimental Results

We present the summary of our 12 experimental results in the following figures. Due to the size of the tables, they are not included in the text but can be viewed via our submission. Each figure shows the validation and test accuracy for the different tasks and models, as well as the training time and number of parameters.

4. Discussions

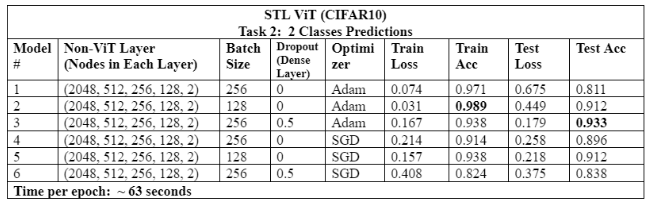
We investigated the performance of multi-task learning (MTL) Vision Transformers (ViTs) and single-task learning (STL) ViTs and Convolutional Neural Networks (CNNs) on two benchmark image classification datasets: CIFAR10 and CIFAR100. Our results address three research questions, which we discuss below.



STL ViT (CIFAR10)
Task 1: 10 Classes Predictions

Model #	Non-ViT Layer (Nodes in Each Layer)	Batch Size	Dropout (Dense Layer)	Optimizer	Train Loss	Train Acc	Test Loss	Test Acc
1	(2048, 1024, 512, 256, 128, 8)	256	0	Adam	0.027	**0.991**	2.153	0.641
2	(2048, 1024, 512, 256, 128, 8)	128	0	Adam	0.042	0.987	2.076	0.657
3	(2048, 1024, 512, 256, 128, 8)	256	0.5	Adam	0.950	0.684	1.015	**0.659**
4	(2048, 1024, 512, 256, 128, 8)	256	0	SGD	0.567	0.805	2.714	0.466
5	(2048, 1024, 512, 256, 128, 8)	128	0	SGD	0.059	0.986	2.747	0.535
6	(2048, 1024, 512, 256, 128, 8)	256	0.5	SGD	1.901	0.262	2.026	0.268
Time per epoch: ~ 63 seconds								

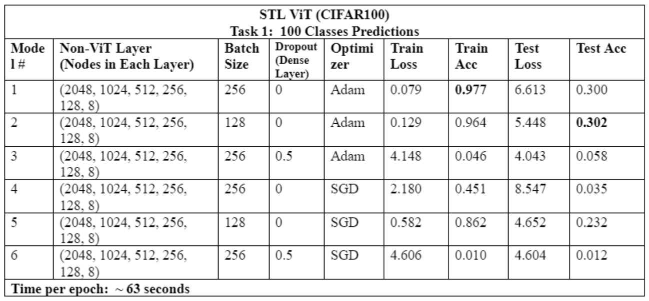
Figure 3. Single-Task Learning on CIFAR10 with 20 epochs using ViT for the first task. The Jupyter Notebook can be found here.



STL ViT (CIFAR10)
Task 2: 2 Classes Predictions

Model #	Non-ViT Layer (Nodes in Each Layer)	Batch Size	Dropout (Dense Layer)	Optimizer	Train Loss	Train Acc	Test Loss	Test Acc
1	(2048, 512, 256, 128, 2)	256	0	Adam	0.074	0.971	0.675	0.811
2	(2048, 512, 256, 128, 2)	128	0	Adam	0.031	**0.989**	0.449	0.912
3	(2048, 512, 256, 128, 2)	256	0.5	Adam	0.167	0.938	0.179	**0.933**
4	(2048, 512, 256, 128, 2)	256	0	SGD	0.214	0.914	0.258	0.896
5	(2048, 512, 256, 128, 2)	128	0	SGD	0.157	0.938	0.218	0.912
6	(2048, 512, 256, 128, 2)	256	0.5	SGD	0.408	0.824	0.375	0.838
Time per epoch: ~ 63 seconds								

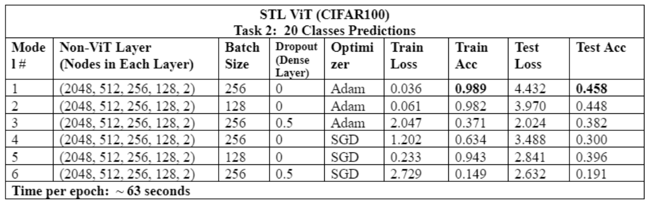
Figure 4. Single-Task Learning on CIFAR10 with 20 epochs using ViT for the second task. The Jupyter Notebook can be found here.



STL ViT (CIFAR100)
Task 1: 100 Classes Predictions

Model #	Non-ViT Layer (Nodes in Each Layer)	Batch Size	Dropout (Dense Layer)	Optimizer	Train Loss	Train Acc	Test Loss	Test Acc
1	(2048, 1024, 512, 256, 128, 8)	256	0	Adam	0.079	**0.977**	6.613	0.300
2	(2048, 1024, 512, 256, 128, 8)	128	0	Adam	0.129	0.964	5.448	**0.302**
3	(2048, 1024, 512, 256, 128, 8)	256	0.5	Adam	4.148	0.046	4.043	0.058
4	(2048, 1024, 512, 256, 128, 8)	256	0	SGD	2.180	0.451	8.547	0.035
5	(2048, 1024, 512, 256, 128, 8)	128	0	SGD	0.582	0.862	4.652	0.232
6	(2048, 1024, 512, 256, 128, 8)	256	0.5	SGD	4.606	0.010	4.604	0.012
Time per epoch: ~ 63 seconds								

Figure 5. Single-Task Learning on CIFAR100 with 20 epochs using ViT for the first task. The Jupyter Notebook can be found here.



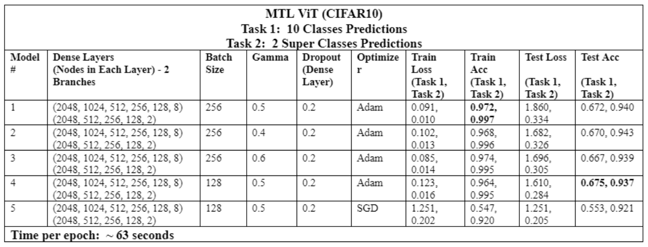
STL ViT (CIFAR100)
Task 2: 20 Classes Predictions

Model #	Non-ViT Layer (Nodes in Each Layer)	Batch Size	Dropout (Dense Layer)	Optimizer	Train Loss	Train Acc	Test Loss	Test Acc
1	(2048, 512, 256, 128, 2)	256	0	Adam	0.036	**0.989**	4.432	**0.458**
2	(2048, 512, 256, 128, 2)	128	0	Adam	0.061	0.982	3.970	0.448
3	(2048, 512, 256, 128, 2)	256	0.5	Adam	2.047	0.371	2.024	0.382
4	(2048, 512, 256, 128, 2)	256	0	SGD	1.202	0.634	3.488	0.300
5	(2048, 512, 256, 128, 2)	128	0	SGD	0.233	0.943	2.841	0.396
6	(2048, 512, 256, 128, 2)	256	0.5	SGD	2.729	0.149	2.632	0.191
Time per epoch: ~ 63 seconds								

Figure 6. Single-Task Learning on CIFAR100 with 20 epochs using ViT for the second task. The Jupyter Notebook can be found here

4.1. Research Question 1: Can an MTL ViT outperform an MTL Convolutional model (ResNet152)?

From Tables 7, 13, 8, and 13, we can see that MTL ViT outperformed MTL ResNet-152 on CIFAR100, while MTL ResNet-152 outperformed MTL ViT on CIFAR10 in terms of testing accuracies. This result suggests that the MTL ViT is better suited for complex classification tasks, as CI-

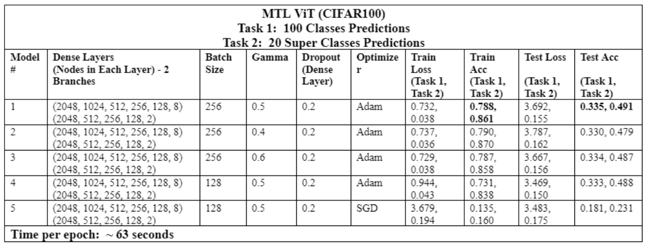


MTL ViT (CIFAR10)
Task 1: 10 Classes Predictions
Task 2: 2 Super Classes Predictions

Model #	Dense Layers (Nodes in Each Layer) - 2 Branches	Batch Size	Gamma	Dropout (Dense Layer)	Optimizer	Train Loss (Task 1, Task 2)	Train Acc (Task 1, Task 2)	Test Loss (Task 1, Task 2)	Test Acc (Task 1, Task 2)
1	(2048, 1024, 512, 256, 128, 8) (2048, 512, 256, 128, 2)	256	0.5	0.2	Adam	0.091, 0.010	**0.972, 0.997**	1.860, 0.334	0.672, 0.940
2	(2048, 1024, 512, 256, 128, 8) (2048, 512, 256, 128, 2)	256	0.4	0.2	Adam	0.102, 0.013	0.968, 0.996	1.682, 0.326	0.670, 0.943
3	(2048, 1024, 512, 256, 128, 8) (2048, 512, 256, 128, 2)	256	0.6	0.2	Adam	0.085, 0.014	0.974, 0.995	1.696, 0.305	0.667, 0.939
4	(2048, 1024, 512, 256, 128, 8) (2048, 512, 256, 128, 2)	128	0.5	0.2	Adam	0.123, 0.016	0.964, 0.995	1.610, 0.284	**0.675, 0.937**
5	(2048, 1024, 512, 256, 128, 8) (2048, 512, 256, 128, 2)	128	0.5	0.2	SGD	1.251, 0.202	0.547, 0.920	1.251, 0.205	0.553, 0.921

Time per epoch: ~ 63 seconds

Figure 7. Multi-Task Learning on CIFAR10 with 20 epochs using ViT for the both tasks. The Jupyter Notebook can be found here

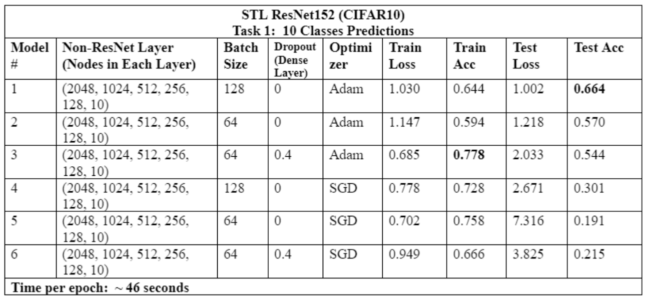


MTL ViT (CIFAR100)
Task 1: 100 Classes Predictions
Task 2: 20 Super Classes Predictions

Model #	Dense Layers (Nodes in Each Layer) - 2 Branches	Batch Size	Gamma	Dropout (Dense Layer)	Optimizer	Train Loss (Task 1, Task 2)	Train Acc (Task 1, Task 2)	Test Loss (Task 1, Task 2)	Test Acc (Task 1, Task 2)
1	(2048, 1024, 512, 256, 128, 8) (2048, 512, 256, 128, 2)	256	0.5	0.2	Adam	0.732, 0.038	**0.788, 0.861**	3.692, 0.155	**0.335, 0.491**
2	(2048, 1024, 512, 256, 128, 8) (2048, 512, 256, 128, 2)	256	0.4	0.2	Adam	0.737, 0.036	0.790, 0.870	3.787, 0.162	0.330, 0.479
3	(2048, 1024, 512, 256, 128, 8) (2048, 512, 256, 128, 2)	256	0.6	0.2	Adam	0.729, 0.038	0.787, 0.858	3.667, 0.156	0.334, 0.487
4	(2048, 1024, 512, 256, 128, 8) (2048, 512, 256, 128, 2)	128	0.5	0.2	Adam	0.944, 0.043	0.731, 0.838	3.469, 0.150	0.333, 0.488
5	(2048, 1024, 512, 256, 128, 8) (2048, 512, 256, 128, 2)	128	0.5	0.2	SGD	3.679, 0.194	0.135, 0.160	3.483, 0.175	0.181, 0.231

Time per epoch: ~ 63 seconds

Figure 8. Multi-Task Learning on CIFAR100 with 20 epochs using ViT for the both tasks. The Jupyter Notebook can be found here

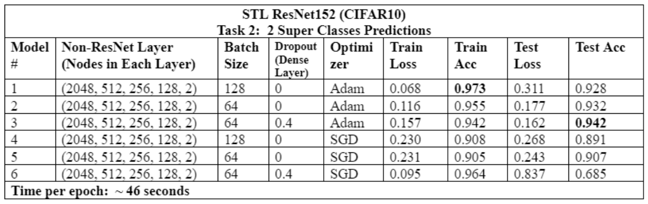


STL ResNet152 (CIFAR10)
Task 1: 10 Classes Predictions

Model #	Non-ResNet Layer (Nodes in Each Layer)	Batch Size	Dropout (Dense Layer)	Optimizer	Train Loss	Train Acc	Test Loss	Test Acc
1	(2048, 1024, 512, 256, 128, 10)	128	0	Adam	1.030	0.644	1.002	**0.664**
2	(2048, 1024, 512, 256, 128, 10)	64	0	Adam	1.147	0.594	1.218	0.570
3	(2048, 1024, 512, 256, 128, 10)	64	0.4	Adam	0.685	**0.778**	2.033	0.544
4	(2048, 1024, 512, 256, 128, 10)	128	0	SGD	0.778	0.728	2.671	0.301
5	(2048, 1024, 512, 256, 128, 10)	64	0	SGD	0.702	0.758	7.316	0.191
6	(2048, 1024, 512, 256, 128, 10)	64	0.4	SGD	0.949	0.666	3.825	0.215

Time per epoch: ~ 46 seconds

Figure 9. Single-Task Learning on CIFAR10 with 20 epochs using ResNet152 for the first task. The Jupyter Notebook can be found here



STL ResNet152 (CIFAR10)
Task 2: 2 Super Classes Predictions

Model #	Non-ResNet Layer (Nodes in Each Layer)	Batch Size	Dropout (Dense Layer)	Optimizer	Train Loss	Train Acc	Test Loss	Test Acc
1	(2048, 512, 256, 128, 2)	128	0	Adam	0.068	**0.973**	0.311	0.928
2	(2048, 512, 256, 128, 2)	64	0	Adam	0.116	0.955	0.177	0.932
3	(2048, 512, 256, 128, 2)	64	0.4	Adam	0.157	0.942	0.162	**0.942**
4	(2048, 512, 256, 128, 2)	128	0	SGD	0.230	0.908	0.268	0.891
5	(2048, 512, 256, 128, 2)	64	0	SGD	0.231	0.905	0.243	0.907
6	(2048, 512, 256, 128, 2)	64	0.4	SGD	0.095	0.964	0.837	0.685

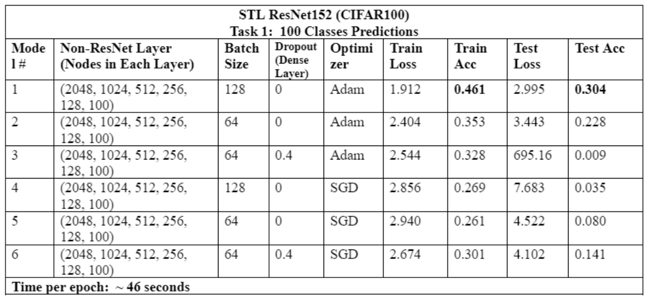
Time per epoch: ~ 46 seconds

Figure 10. Single-Task Learning on CIFAR10 with 20 epochs using ResNet152 for the second task. The Jupyter Notebook can be found here

FAR100 is a more complex dataset than CIFAR10. With more complex datasets or longer training epochs, we expect MTL ViT to outperform MTL ResNet-152 on both datasets.

4.2. Research Question 2: Can an MTL ViT outperform two STL ViTs?

From Tables 7, 8, 3, 4, 5, and 6, we can see that MTL ViT outperformed two STL ViTs on both CIFAR10 and CIFAR100 in terms of testing accuracies. This result aligns with previous studies on benchmark datasets such as Taskonomy, Replica, and CocoDoom [13, 12, 9]. We believe that the superior performance of MTL ViT is due to the sharing of the same backbone between the two tasks, which enables the network to learn more representations while significantly reducing the number of parameters.

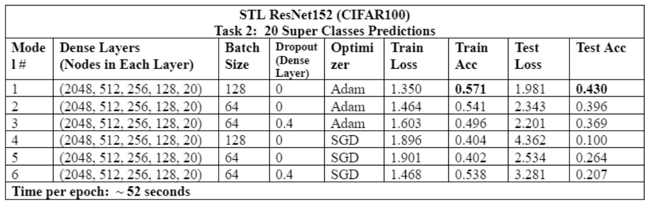


STL ResNet152 (CIFAR100)
Task 1: 100 Classes Predictions

Model #	Non-ResNet Layer (Nodes in Each Layer)	Batch Size	Dropout (Dense Layer)	Optimizer	Train Loss	Train Acc	Test Loss	Test Acc
1	(2048, 1024, 512, 256, 128, 100)	128	0	Adam	1.912	**0.461**	2.995	**0.304**
2	(2048, 1024, 512, 256, 128, 100)	64	0	Adam	2.404	0.353	3.443	0.228
3	(2048, 1024, 512, 256, 128, 100)	64	0.4	Adam	2.544	0.328	695.16	0.009
4	(2048, 1024, 512, 256, 128, 100)	128	0	SGD	2.856	0.269	7.683	0.035
5	(2048, 1024, 512, 256, 128, 100)	64	0	SGD	2.940	0.261	4.522	0.080
6	(2048, 1024, 512, 256, 128, 100)	64	0.4	SGD	2.674	0.301	4.102	0.141

Time per epoch: ~ 46 seconds

Figure 11. Single-Task Learning on CIFAR100 with 20 epochs using ResNet152 for the first task. The Jupyter Notebook can be found here

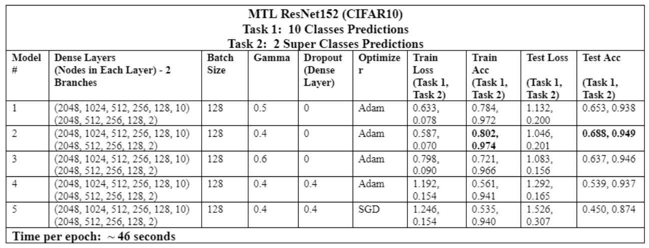


STL ResNet152 (CIFAR100)
Task 2: 20 Super Classes Predictions

Model #	Dense Layers (Nodes in Each Layer)	Batch Size	Dropout (Dense Layer)	Optimizer	Train Loss	Train Acc	Test Loss	Test Acc
1	(2048, 512, 256, 128, 20)	128	0	Adam	1.350	**0.571**	1.981	**0.430**
2	(2048, 512, 256, 128, 20)	64	0	Adam	1.464	0.541	2.343	0.396
3	(2048, 512, 256, 128, 20)	64	0.4	Adam	1.603	0.496	2.201	0.369
4	(2048, 512, 256, 128, 20)	128	0	SGD	1.896	0.404	4.362	0.100
5	(2048, 512, 256, 128, 20)	64	0	SGD	1.901	0.402	2.534	0.264
6	(2048, 512, 256, 128, 20)	64	0.4	SGD	1.468	0.538	3.281	0.207

Time per epoch: ~ 52 seconds

Figure 12. Single-Task Learning on CIFAR100 with 20 epochs using ResNet152 for the second task. The Jupyter Notebook can be found here

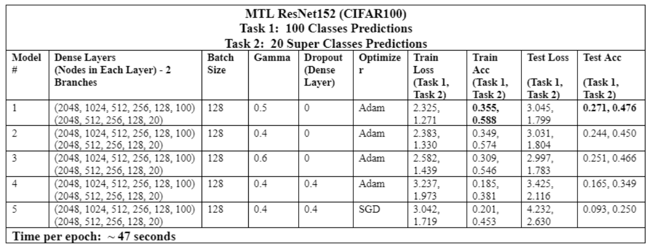


MTL ResNet152 (CIFAR10)
Task 1: 10 Classes Predictions
Task 2: 2 Super Classes Predictions

Model #	Dense Layers (Nodes in Each Layer) - 2 Branches	Batch Size	Gamma	Dropout (Dense Layer)	Optimizer	Train Loss (Task 1, Task 2)	Train Acc (Task 1, Task 2)	Test Loss (Task 1, Task 2)	Test Acc (Task 1, Task 2)
1	(2048, 1024, 512, 256, 128, 10) (2048, 512, 256, 128, 2)	128	0.5	0	Adam	0.633, 0.078	0.784, 0.972	1.132, 0.200	0.653, 0.938
2	(2048, 1024, 512, 256, 128, 10) (2048, 512, 256, 128, 2)	128	0.4	0	Adam	0.587, 0.070	**0.802, 0.974**	1.046, 0.201	**0.688, 0.949**
3	(2048, 1024, 512, 256, 128, 10) (2048, 512, 256, 128, 2)	128	0.6	0	Adam	0.798, 0.090	0.721, 0.966	1.083, 0.156	0.637, 0.946
4	(2048, 1024, 512, 256, 128, 10) (2048, 512, 256, 128, 2)	128	0.4	0.4	Adam	1.192, 0.154	0.561, 0.941	1.292, 0.165	0.539, 0.937
5	(2048, 1024, 512, 256, 128, 10) (2048, 512, 256, 128, 2)	128	0.4	0.4	SGD	1.246, 0.154	0.535, 0.940	1.526, 0.307	0.450, 0.874

Time per epoch: ~ 46 seconds

Figure 13. Multi-Task Learning on CIFAR10 with 20 epochs using ResNet152 for the both tasks. The Jupyter Notebook can be found here



MTL ResNet152 (CIFAR100)
Task 1: 100 Classes Predictions
Task 2: 20 Super Classes Predictions

Model #	Dense Layers (Nodes in Each Layer) - 2 Branches	Batch Size	Gamma	Dropout (Dense Layer)	Optimizer	Train Loss (Task 1, Task 2)	Train Acc (Task 1, Task 2)	Test Loss (Task 1, Task 2)	Test Acc (Task 1, Task 2)
1	(2048, 1024, 512, 256, 128, 100) (2048, 512, 256, 128, 20)	128	0.5	0	Adam	2.325, 1.271	**0.355, 0.588**	3.045, 1.799	**0.271, 0.476**
2	(2048, 1024, 512, 256, 128, 100) (2048, 512, 256, 128, 20)	128	0.4	0	Adam	2.383, 1.330	0.349, 0.574	3.031, 1.804	0.244, 0.450
3	(2048, 1024, 512, 256, 128, 100) (2048, 512, 256, 128, 20)	128	0.6	0	Adam	2.582, 1.439	0.309, 0.546	2.997, 1.783	0.251, 0.466
4	(2048, 1024, 512, 256, 128, 100) (2048, 512, 256, 128, 20)	128	0.4	0.4	Adam	3.237, 1.973	0.185, 0.381	3.425, 2.116	0.165, 0.349
5	(2048, 1024, 512, 256, 128, 100) (2048, 512, 256, 128, 20)	128	0.4	0.4	SGD	3.042, 1.719	0.201, 0.453	4.232, 2.630	0.093, 0.250

Time per epoch: ~ 47 seconds

Figure 14. Multi-Task Learning on CIFAR100 with 20 epochs using ResNet152 for the both tasks. The Jupyter Notebook can be found here

4.3. Answer to Research Question 3: Can an STL ViT model outperform an STL Convolution-based model (ResNet-152)?

For this question, there is no clear answer. From the tables 9 and 3, we can see that ResNet-152 slightly outperformed ViT on STL for CIFAR10 on the 10-class classification task. Similarly, from the tables 10 and 4, ResNet-152 slightly outperformed ViT on STL for CIFAR10 on the 2-superclass classification task. However, in the tables 11 and 5, ViT slightly outperformed ResNet-152 on STL for CIFAR10 on the 2-superclass classification task.

Moreover, from the tables 12 and 6, we can observe that ViT slightly outperformed ResNet-152 on STL for CIFAR10 on the 2-superclass classification task.

Suppose we use a more complex dataset or train with more epochs, we might see that ViT outperforms ResNet-152 in both CIFAR10 and CIFAR100. ViT can capture global features due to its positional embedding and attention mechanism, while ResNet-152 can capture local features due to the convolutional operation.

5. Contributions

This project was a collaborative effort between two authors, with equal contribution from each. Minh was responsible for coding, analyzing, and collecting results for both STL and MTL experiments using the ViT architecture, as shown in Tables 3, 4, 5, 6, 7, and 8. Peter, on the other hand, handled the same tasks for both STL and MTL experiments using the ResNet-152 architecture, as shown in Tables 9, 10, 11, 12, 13, and 14. Each author conducted their work independently. The authors declare that they have no conflicts of interest.

Acknowledgements. We would like to express our gratitude to Professor Jianhua (Jason) Xuan from the ECE Department at Virginia Tech for his valuable comments, corrections, and inspiration.

References

[1] Y. Chen, J. Yu, Y. Zhao, J. Chen, and X. Du. Task's choice: Pruning-based feature sharing (pbfs) for multi-task learning. *Entropy*, 24(3):432, 2022.

[2] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele. The cityscapes dataset for semantic urban scene understanding. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3213–3223, 2016.

[3] M. Crawshaw. Multi-task learning with deep neural networks: A survey. *arXiv preprint arXiv:2009.09796*, 2020.

[4] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[5] Evannex. Andrej karpathy talks tesla autopilot amp; multi-task learning, Aug 2019.

[6] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[7] A. Krizhevsky, V. Nair, and G. Hinton. Cifar-10 and cifar-100 datasets. *URl: https://www. cs. toronto. edu/kriz/cifar. html*, 6(1):1, 2009.

[8] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft coco: Common objects in context. In *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13*, pages 740–755. Springer, 2014.

[9] A. Mahendran, H. Bilen, J. F. Henriques, and A. Vedaldi. Researchdoom and cocodoom: Learning computer vision with games. *arXiv preprint arXiv:1610.02431*, 2016.

[10] I. Misra, A. Shrivastava, A. Gupta, and M. Hebert. Cross-stitch networks for multi-task learning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3994–4003, 2016.

[11] S. Ruder. An overview of multi-task learning in deep neural networks. *arXiv preprint arXiv:1706.05098*, 2017.

[12] J. Straub, T. Whelan, L. Ma, Y. Chen, E. Wijmans, S. Green, J. J. Engel, R. Mur-Artal, C. Ren, S. Verma, et al. The replica dataset: A digital replica of indoor spaces. *arXiv preprint arXiv:1906.05797*, 2019.

[13] A. R. Zamir, A. Sax, W. Shen, L. J. Guibas, J. Malik, and S. Savarese. Taskonomy: Disentangling task transfer learning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3712–3722, 2018.

[14] X. Zheng, B. Wu, X. Zhu, and X. Zhu. Multi-task deep learning seismic impedance inversion optimization based on homoscedastic uncertainty. *Applied Sciences*, 12(3):1200, 2022.