Transfer Learning for Identifying Land Use and Land Cover from Satellite Imagery

Chenchen Gu
Stanford University
cygu@stanford.edu

Abstract

Land use and land cover data is valuable for many applications, such as environmental monitoring and urban planning. In this project, we apply modern deep learning and computer vision methods to identify land use and land cover from satellite imagery. We apply transfer learning using pre-trained convolutional neural networks and vision transformers. After fine-tuning on the NWPU-RESISC45 dataset, which contains 31,500 RGB satellite images from 45 land use and land cover classes, we achieve 95.9% classification accuracy on the unseen test set. We find that transfer learning with full fine-tuning outperforms transfer learning with feature extraction only. In addition, we find that the pre-training dataset and methods can have a significant impact on the model's ability to transfer to new tasks.

1. Introduction

Land use describes how land is being by humans, such as for residential, commercial, industrial, or agricultural purposes. Land cover is a closely related concept, describing the physical components present and visible on the surface of the Earth, such as forests, wetlands, and urban areas. Identifying and tracking land use and land cover is valuable for many applications. It is important for monitoring and managing changes in the environment, such as deforestation, habitat loss, and biome changes due to climate change [34]. Urban and regional planning relies on land use data for zoning decisions, infrastructure development, and resource allocation [2]. Land cover data is valuable for agricultural management to assess the health of croplands and perform crop rotation, irrigation planning, and soil management.

Collecting land use and land cover data from ground surveys and field surveys can capture detailed information, but can be costly and slow to collect. Instead, satellite imagery and remote sensing can be used to effectively collect land use and land cover data at a large scale. Nearly all the Earth's surface has been captured via satellite imagery, with large-scale satellite imagery collection missions such as the Landsat program run by NASA and USGS [23] and the Sentinel-2 program operated by the European Space Agency [9]. As a result, there is far too much satellite imagery data to be analyzed and labeled completely by humans.

In this project, we apply modern deep learning and computer vision methods to identify land use and land cover from satellite imagery. By doing so, large amounts of such data can be efficiently and accurately collected, which would enable frequent and up-to-date large-scale monitoring. Precisely, the input to our algorithm is an RGB satellite image. We then use convolutional neural networks (CNNs) and vision transformer (ViTs) to output a predicted land use/land cover class, e.g., forest, farmland, freeway, etc. We apply transfer learning using pre-trained ResNet-50 [13] and ViT-B/16 [8] models. After fine-tuning, we are able to achieve 95.9% classification accuracy on the unseen test set of the NWPU-RESISC45 dataset, which contains 31,500 satellite images from 45 classes.

2. Related work

Satellite imagery datasets Satellite imagery has been of interest to the machine learning community for many years. One early dataset is the 2010 UC Merced Land Use dataset [38], which contains 100 satellite images from each of 21 land use classes, such as agricultural, forest, parking lot, etc. Yang and Newsam, the dataset authors, apply a bag-of-visual-words approach to classify land use [38]. However, the UC Merced Land Use dataset is too small for modern deep learning methods in computer vision. Since then, larger datasets for satellite image classification have been constructed. Some datasets classify specific, narrow attributes of land, such as FireRisk [28] for fire risk assessment and CropHarvest [33] for agricultural classification. Other datasets classify broader categories of land use. EuroSat [14] contains 27,000 satellite images from 10 classes, BigEarthNet [30] contains 590,000 images annotated with multiple labels each, and Functional Map of the

World [6] contains over 1 million satellite images annotated with bounding boxes that show land or building use from among 63 categories, along with geospatial metadata for each image. In this project, we use the NWPU-RESISC45 dataset [5] dataset, which contains 31,500 satellite images from 45 scene classes. We choose NWPU-RESISC45 for its tractable size for this project and its diversity of scene classes. We discuss more details in §3.

Deep learning computer vision models Deep learning has been extremely popular in computer vision and has been successfully applied to many tasks and domains. One popular and powerful class of models are convolutional neural networks (CNNs), which use convolutions to efficiently perform localized operations to images and intermediate hidden states. There are many specific variations and CNN architectures, such as AlexNet [19], GoogLeNet [31], VGGNet [29], ResNet [13], and DenseNet [15]. In this project, we use ResNet, which uses residual shortcut connections to improve gradient flow and training stability in very deep networks. More recently, several computer vision models using the self-attention-based Transformer [35] architecture have been developed, including Vision Transformer (ViT) [8], Swin Transformer [20], and data efficient image Transformers (DeiT) [32]. In this project, we use ViT, which closely resembles the original Transformer architecture.

Transfer learning In transfer learning, a model is first pre-trained on a large dataset, then used to extract features or be fully fine-tuned on a smaller, task-specific dataset of interest. Early transfer learning approaches found that passing features from CNNs pre-trained on ImageNet [26] into simple linear SVM or logistic regression classifiers could achieve high performance on downstream tasks and challenges [7, 27, 39]. Transfer learning has been successfully applied to many models and architectures. For example, performing transfer learning by fully fine-tuning a pre-trained ResNet [4] or ViT [8] has been found to achieve strong performance on downstream tasks.

Transfer learning has also been applied to the satellite imagery domain. For example, Jean et al. (2016) [17] fine-tune ImageNet pre-trained CNNs on proxy metrics for poverty in order to predict poverty levels from satellite imagery from five African countries. Several works have investigated training pre-trained models specifically for transfer learning to satellite imagery tasks. Neumann et al. (2019) [24] investigate in-domain representation learning to train features for satellite imagery and which dataset characteristics are beneficial for this task. Wang et al. (2023) [37] apply self-supervised pre-training methods, such as momentum contrast (MoCo) [11], on large-scale unlabeled satellite imagery to train models which can be effectively fine-tuned on downstream satellite imagery applications.

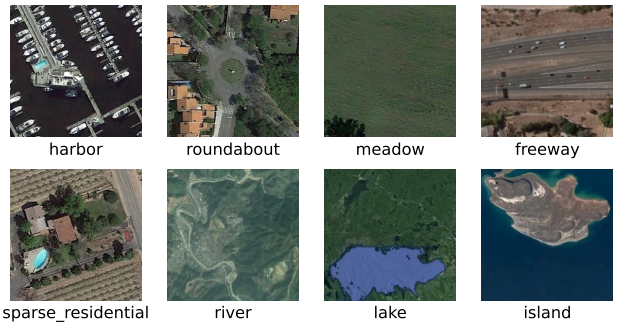

Figure 1. Randomly selected images from the train split of the NWPU-RESISC45 dataset.

Bastani et al. (2023) [3] also pre-train on satellite imagery, but they perform supervised learning on a variety of labeled tasks.

3. Dataset

We use the NWPU-RESISC45 dataset [5] for remote sensing image scene classification (RESISC), created by Northwestern Polytechnical University (NWPU). NWPU-RESISC45 contains 31,500 satellite images from 45 scene classes, with 700 images in each class. We load the dataset from the Hugging Face Hub[1], which has pre-divided the dataset into train, validation, and test splits. The train split contains 60% (18,900 images) of the images in the entire dataset, the validation split contains 20% (6,300 images), and the test split contains 20% (6,300 images).

Figure 1 shows images from NWPU-RESISC45. The 45 scene classes include land use and land cover classes (e.g., desert, farmland, industrial area, mountain), man-made objects and structures (e.g., airplane, bridge, stadium), and natural landscape features (e.g., beach, cloud, river). The images are RGB satellite images with a resolution of 256×256 pixels. The spatial resolution varies from about 30 meters to 0.2 meters per pixels. The dataset was constructed by experts in remote sensing image interpretation, who extracted and labeled satellite imagery from Google Maps. The images span over 100 countries across the world, including least developed, developing, and high developed countries.

NWPU-RESISC45 is designed to be challenging due to high intraclass diversity and high interclass similarity. For interclass diversity, images were carefully selected to display high variability between images, including different illumination conditions, spatial resolution, background, season, and weather conditions. For interclass similarity, some similar classes were chosen that may be hard for a model to distinguish, such as freeway vs. overpass, circular farmland vs. rectangular farmland, and medium residential vs. sparse

[1] https://huggingface.co/datasets/timm/resisc45

residential.

3.1. Data preprocessing

For both training and inference, we perform the standard normalization of subtracting the per-channel (RGB) mean pixel value in the train split and dividing by the per-channel standard deviation of pixel values in the train split. As a result, the distribution of each RGB channel in the training set has mean 0 and standard deviation 1, which allows for more stable and efficient training.

The sizes of the images in NWPU-RESISC45 are 256×256 pixels, but the models we use in the experiments expect an input size of 224×224 pixels. During training, to resize the input and as a form of data augmentation, we randomly select a 224×224 sized crop from the original 256×256 sized image (a new random crop is selected each time the image is seen across epochs). After selecting the crop, for additional data augmentation, we randomly flip the image horizontally and vertically, independently with probability 0.5 each. Flipping horizontally or vertically is a valid form of data augmentation for this dataset since the satellite images are taken aerially from a vertical viewpoint. During inference, we simply resize the entire image to 224×224 pixels using bilinear interpolation and anti-aliasing.

4. Methods

4.1. Problem setup

Concretely, the input x to our problem is an RGB satellite image, as described in §3. These images are represented as tensors with pixel values in the range $[0, 255]$. After performing data normalization and resizing, the images become $3 \times 224 \times 224$ tensors of real numbers.

The output $\hat{y}$ to our problem is a predicted scene class for the image, out of the K scene classes defined in the dataset. More specifically, the model outputs a set of logits/scores z, where each $z_i \in \mathbb{R}$ is the logit value for class $i \in [1, K]$. These logits are converted into a probability distribution via the softmax function. The probability p_i that the model puts on class i is given by

$$p_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}. \tag{1}$$

The model's prediction is taken to be the class with the highest probability. The higher the probability, the more "confident" the model is in its prediction.

During training, we use the standard cross-entropy loss function between the model's predictions and the ground-truth correct label. Letting y be the ground-truth correct class label for an example, the loss on that example is given by

$$\mathcal{L}(\theta) = -\log(p_y) \tag{2}$$

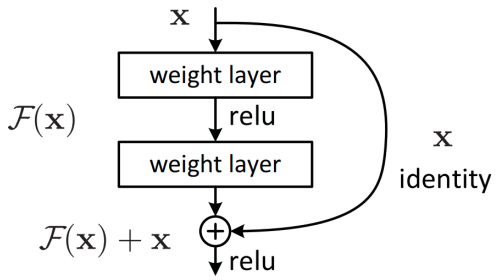

Figure 2. Building block for residual learning. The shortcut connection implements an identity mapping which skips over the stacked layers. Figure is taken from He et al. (2016) [13]

using the natural logarithm with base e, and θ is the model's parameters that we are trying to optimize. If the model incorrectly places close to 0 probability on the correct class y, the loss approaches $+\infty$, and if the model correctly places close to probability 1 on the correct class y, the loss approaches 0. To aggregate the losses of examples across a mini-batch or the entire dataset, we simply average the losses.

4.2. Models

We use two model architectures for our experiments: 1) ResNet [13], which is a convolutional neural network (CNN) architecture, and 2) the vision transformer (ViT) [8].

4.2.1 ResNet

ResNet, short for residual network, is a CNN architecture proposed by He et al. in 2015 [13]. At the time of its release, ResNet significantly advanced the state-of-the-art in computer vision, winning first place in the 2015 ImageNet Large Scale Visual Recognition Challenge (ILSVRC), which measures error on the ImageNet test set [26].

ResNet seeks to address the observed issue that deeper neural networks are more difficult to train due to vanishing or exploding gradients, which are propagated and amplified through the model's layers. To alleviate this issue, ResNet introduces a residual learning framework, where layers learn residual functions with respect to the layer inputs, rather than learn unreferenced functions. To do so, a shortcut connection, which implements the identity mapping, is applied to every few stacked layers, forming a building block, as shown in Figure 2.

Formally, a building block is defined as

$$\mathbf{y} = \mathcal{F}(\mathbf{x}, \{W_i\}) + \mathbf{x}. \tag{3}$$

In this equation, $\mathbf{x}$ and $\mathbf{y}$ are the input and output vectors of the stacked layers, respectively. The function $\mathcal{F}(\mathbf{x}, \{W_i\})$ is the function to be learned by the stacked

layers, parametrized by weights $\{W_i\}$. The elementwise addition of the input $\mathbf{x}$ to this function is the identity mapping shortcut connection. As a result of this shortcut connection, the stacked layers learn an offset or residual from the identity mapping, rather than learning a completely new function from scratch. These shortcut connections allow for improved gradient flow throughout the layers of a deep neural network, allowing for more stable training and better overall results [13].

Concretely, in the ResNet CNN architecture, the building blocks consist of shortcut connections applied to stacks of convolution layers. In our project, we use the 50-layer ResNet-50 architecture, which uses bottleneck residual blocks consisting of three layers: 1×1, 3×3, and 1×1 convolutions. Batch normalization [16] is applied after each convolution. Within the 50 layers, these blocks are repeated 3–6 times before the output width and height are both halved, and the number of channels is doubled. After all the convolutional layers, average pooling, a fully connected layer, and softmax is applied to obtain classification probabilities. ResNet-50 contains approximately 26 million parameters.

4.2.2 Vision Transformer (ViT)

The Vision Transformer (ViT) model architecture [8] applies the self-attention-based Transformer architecture [35] to image classification tasks.

The self-attention layer takes a matrix of input vectors $X \in \mathbb{R}^{n \times d}$, where n is the number of input tokens and d is the embedding dimensionality. It is parametrized by weight matrices $W_Q, W_K, W_V \in \mathbb{R}^{d \times d_k}$, where d_k is the dimensionality of the queries, keys, and values. Then, the query, key, and value matrices ($Q, K, V \in \mathbb{R}^{n \times d_k}$) are computed as

$$Q = XW_Q, \quad K = XW_K, \quad V = XW_V. \tag{4}$$

Then, scaled dot-product attention is used to compute the output matrix $O \in \mathbb{R}^{n \times d_k}$ as

$$O = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right) V. \tag{5}$$

In multi-head self-attention, there are h attention heads, each with their own weight matrices. Often, the attention dimensionality is set to be $d_k = d/h$. Each head computes an output matrix $O_h \in \mathbb{R}^{n \times d_k}$. Then, to combine these outputs into a final output $O \in \mathbb{R}^{n \times d}$, they are concatenated together and multiplied by a matrix $W_O \in \mathbb{R}^{(h \cdot d_k) \times d}$, i.e.,

$$O = [O_1; O_2; \ldots; O_h]\, W_O. \tag{6}$$

Transformer encoder blocks consist of multi-head self-attention and multilayer perceptron (MLP) blocks, along with layer normalization [1] and additive residual shortcut connections, as discussed earlier [13].

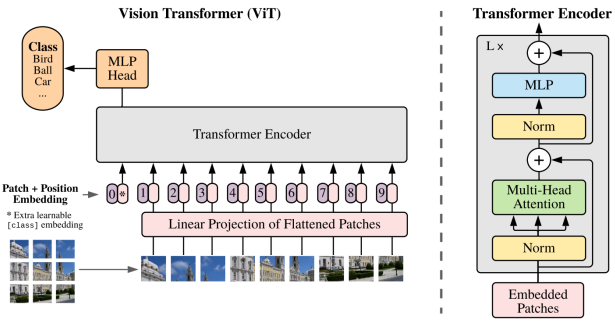

Figure 3. Vision Transformer (ViT) model architecture diagram, taken from Dosovitskiy et al. (2020) [8]. Input images are tokenized into patches, which are then passed into the Transformer encoder. To classify the image, an MLP head is applied to the final hidden state of the special `[class]` token prepended to the beginning of the sequence.

The Vision Transformer (ViT), shown in Figure 3, first divides the input image into patches, linearly projects the flattened patches, and adds position encodings to generate input tokens. The position encoding allows the model to know where the original patches appeared, since the self-attention layer does not explicitly contain any information about position. Then, these tokens are passed through the Transformer encoder. To classify the image, an MLP head is applied to the final hidden state of the special `[class]` token prepended to the beginning of the sequence.

In this project, we use the ViT-B/16 architecture, which is ViT-Base with 16×16 input patches. ViT-B/16 has approximately 87 million parameters.

4.3. Baseline: random initialization

As a baseline learning method, we randomly initialize the ResNet-50 and ViT-B/16 models, then train on only the NWPU-RESISC45 dataset using the cross-entropy loss function (Equation 2). We use the default initialization methods used in the PyTorch TorchVision [22] implementations of these models, which is Kaiming initialization [12] for ResNet and Xavier initialization [10] for ViT.

4.4. Method: transfer learning

To improve on the random initialization baseline, we use transfer learning [7, 27, 39, 40],where a model is first pre-trained on a large dataset, then used to extract features or be fully fine-tuned on a smaller, task-specific dataset of interest. In our project, we fine-tune on the relatively small NWPU-RESISC45 dataset. Both ResNet and ViT have been found to enable strong transfer learning performance to downstream tasks when pre-trained on a large dataset [4, 8].

When performing transfer learning for this project, we replace the final fully connected layer of ResNet-50 and

MLP classification head of ViT-B/16 with a new, randomly initialized linear layer mapping from the final hidden state to the 45 classes of the NWPU-RESISC45 dataset. Then, we fine-tune on NWPU-RESISC45 using the cross-entropy loss function (Equation 2). We experiment with two methods for fine-tuning.

1. *Feature extraction*. In this method, the new randomly initialized linear head are the only parameters that are trained, and the rest of the pre-trained model is frozen. In other words, the pre-trained model is used to extract features from the fine-tuning dataset, where the features are the final hidden state. Then, these features are passed as input into the new linear classifier, which is being fine-tuned.

2. *Full fine-tuning*. In this method, the entire pre-trained model and new linear head is fine-tuned on the new dataset. We use a lower learning rate during fine-tuning to avoid moving far away from the pre-trained weights, which are likely a good starting point.

Feature extraction is more computationally efficient to train, since only the new linear classifier is being updated. Feature extraction works well if the fine-tuning dataset is similar to the pre-training dataset, but it may not work as well if the datasets are dissimilar, in which case the pre-trained model may not extract optimal, meaningful features from the new fine-tuning dataset. Full fine-tuning requires more computational resources to train, as the entire pre-trained model is updated, but it may achieve better performance in some cases, since updating all parameters allows for more representational power and may better handle large dissimilarities between the pre-training and fine-tuning datasets.

4.5. Pre-trained models

We experiment with transfer learning using models pre-trained on 1) ImageNet [26] and 2) Sentinel-2 unlabeled satellite imagery data [37].

ImageNet We use ResNet-50 and ViT-B/16 weights uploaded and pre-trained by PyTorch TorchVision [22, 36] on ImageNet [26]. ImageNet is a large-scale object classification dataset containing over one million images belonging to 1,000 classes. The pre-training classification objective is to minimize the cross-entropy loss, as in Equation 2.

Sentinel-2 We use ResNet-50 weights which were pre-trained on over one million unlabeled Sentinel-2 RGB satellite images via self-supervised learning [37]. Specifically, we use the weights that were trained via momentum contrast (MoCo) [11], which is a form of contrastive representation learning. In MoCo, positive pairs are augmentations of the same example, and all other images are negative samples. The negative samples are stored in a running queue of keys, and the contrastive loss is computed between the current query and keys. The gradients are computed only through the queries, and the key encoder is slowly updated via a momentum update rule.

4.6. Existing code libraries

We used the ResNet-50 and ViT-B/16 models implemented by the PyTorch TorchVision library [22], along with pre-trained weights provided by TorchVision. We wrote training code in PyTorch [25], referring to a PyTorch classifier training tutorial[2].

5. Experiments and results

5.1. Training procedure

For each model and method, we fine-tune on the NWPU-RESISC45 dataset for 10 epochs. We use a batch size of 128 examples in order to take advantage of GPU parallelism and maximize memory utilization. We use the widely used AdamW optimizer [18, 21]. AdamW estimates the first and second moments of the gradient to enable per-parameter adaptive learning rates and uses weight decay for regularization. The weight decay λ hyperparameter controls the strength of this regularization. We use the standard AdamW beta hyperparameters of $(\beta_1, \beta_2) = (0.9, 0.999)$. For the learning rate schedule, we perform a linear warmup from 0 to the maximal learning rate over the first 5% of training steps, followed by a linear decay from the maximal learning rate to 0 for the remaining training steps.

We perform a hyperparameter grid search across the maximal learning rate and weight decay strength, training each combination for the full 10 epochs and choosing the combination that attains the highest accuracy on the validation set. The hyperparameter search space and chosen hyperparameters for each model and method are shown in Table 1. The exact search spaces were chosen by observing hyperparameters used in the literature [13, 8] and manually narrowing down search spaces by performing short training runs.

5.2. Results

Our primary evaluation metric is accuracy, defined simply as the proportion of images where the model's top-1 predicted class label matches the ground-truth correct class label. For the chosen hyperparameter combination for each model and method, we report accuracies on the train, validation, and test sets in Table 2.

For both ResNet-50 and ViT-B/16, we find that performing full fine-tuning transfer learning using ImageNet pre-

[2] https://pytorch.org/tutorials/beginner/blitz/cifar10_tutorial.html

Model and method	Maximal learning rate	Weight decay
ResNet-50		
Random initialization	{**1e-3**, 3e-3, 1e-2}	{**1e-4**, 1e-3}
ImageNet pre-train, feature extraction	{1e-3, 3e-3, **1e-2**}	{0, **1e-4**}
ImageNet pre-train, full fine-tuning	{3e-5, 1e-4, **3e-4**}	{1e-4, **1e-3**}
Sentinel-2 pre-train, feature extraction	{1e-3, 3e-3, **1e-2**}	{0, **1e-4**}
Sentinel-2 pre-train, full fine-tuning	{3e-5, 1e-4, **3e-4**}	{1e-4, **1e-3**}
ViT-B/16		
Random initialization	{1e-4, **3e-4**, 1e-3}	{**1e-2**, 1e-1}
ImageNet pre-train, feature extraction	{1e-3, 3e-3, **1e-2**}	{0, **1e-4**}
ImageNet pre-train, full fine-tuning	{3e-5, **1e-4**, 3e-4}	{1e-2, **1e-1**}

Table 1. Learning rate and weight decay hyperparameter search spaces for each model and method. The bolded hyperparameter combinations attained the highest accuracy on the validation set.

	Accuracy		
Model and method	Train	Validation	Test
ResNet-50			
Random initialization	0.838	0.813	0.796
ImageNet pre-train, feature extraction	0.949	0.883	0.881
ImageNet pre-train, full fine-tuning	0.998	0.960	**0.959**
Sentinel-2 pre-train, feature extraction	0.863	0.827	0.820
Sentinel-2 pre-train, full fine-tuning	0.977	0.935	0.930
ViT-B/16			
Random initialization	0.881	0.781	0.753
ImageNet pre-train, feature extraction	0.966	0.901	0.885
ImageNet pre-train, full fine-tuning	**0.999**	**0.961**	0.955

Table 2. Train, validation, and test set accuracies for each model and method, using the training hyperparameters chosen in Table 1. For both ResNet-50 and ViT-B/16, transfer learning with full fine-tuning on the ImageNet pre-trained weights attain the highest performance.

trained weights achieves the highest accuracies. The top accuracies are nearly identical between ResNet-50 and ViT-B/16, with test set accuracies of 0.959 and 0.955, respectively. These high accuracies indicate that transfer learning can successfully be applied to the NWPU-RESISC45 dataset.

We do not believe that these models have overfitted to the training data. Although the training accuracies are nearly perfect, the test set accuracies are very close at around just 0.05 lower. The weight decay in AdamW [21] and our use of random data augmentations and transformations in §3.1 likely contributed to the avoidance of overfitting. It is somewhat surprising that the top ResNet-50 and ViT-B/16 models attain the same accuracies, since ViT-B/16 has over three times as many parameters as ResNet-50 (87 million parameters vs. 26 million parameters).

For both ResNet-50 and ViT-B/16, the baseline random initialization method performs significantly worse than the transfer learning methods. This is expected, as random initialization does not start from capable pre-trained weights that already encode meaningful, useful features. However, random initialization still achieves moderately high, reasonable test set accuracies of 0.796 and 0.753 for ResNet-50 and ViT-B/16, respectively. So, random initialization is not an extremely weak baseline method, it just performs relatively poorly in comparison to the much stronger transfer learning methods.

In both ResNet-50 and ViT-B/16, transfer learning with feature extraction only performs worse than transfer learning with full fine-tuning, with final test set accuracies lower by about 0.07-0.11. We hypothesize that this gap may be due to differences between the pre-training datasets and the fine-tuning NWPU-RESISC45 dataset. The pre-trained weights may not extract optimal features for the new dataset, so allowing the full weights to be updated allows for more flexibility to learn altered features that are more useful specifically for NWPU-RESISC45. However, feature extraction only still achieves relatively high accuracies of

around 0.88 when using the ImageNet pre-trained weights. So, feature extraction is still a reasonable method to use, especially when computational resources are limited. Also, since only the linear classifier head is trained, the same pre-trained backbone weights can be used for many downstream tasks. This allows for efficient task-switching if multiple downstream tasks need to be performed, as only the linear classifier head needs to switched out. This also reduces the size of the weights that need to be stored.

Somewhat surprisingly, transfer learning with the ImageNet pre-trained weights performs better than when using the Sentinel-2 pre-trained weights. When performing full fine-tuning, the ImageNet weights attain a test accuracy of 0.959, whereas the Sentinel-2 weights attain a test accuracy of 0.930. This is surprising because the Sentinel-2 dataset of satellite imagery is more similar to the NWPU-RESISC45 satellite imagery dataset than ImageNet is, which contains ground-based imagery of miscellaneous objects and no satellite imagery [26]. We hypothesize that this performance gap may because the ImageNet weights were pre-trained for longer than the Sentinel-2 weights. The ImageNet weights were pre-trained for 600 epochs [36], whereas the Sentinel-2 weights were pre-trained for 100 epochs [37], with both datasets consisting of around 1 million examples. In this case, the longer pre-training may outweigh the dataset similarity between Sentinel-2 and NWPU-RESISC45. Another potential reason for the difference is that the ImageNet weights are pre-trained via supervised learning on labeled classification data, whereas the Sentinel-2 weights are pre-trained via self-supervised learning on unlabeled data. Since our downstream task is classification, pre-training on a classification task may yield more relevant features. Given more time and resources, future work may investigate whether pre-training on satellite imagery using supervised learning or more data can increase the performance of transfer learning.

5.3. Error analysis

In this section, we perform a qualitative error analysis of which classes the top-performing ResNet-50 model (ImageNet pre-trained, transfer learning, full fine-tuning) predicts incorrectly the most often. Figure 4 displays the test set error rates for all scene classes in NWPU-RESISC45, where the error rate for class y is the proportion of images with true label y that the model incorrectly labels.

The classes with the highest error rates have other classes in the dataset that are very similar. For example, palace and church have the two highest error rates and are very similar to each other from satellite imagery. Runway has the third-highest error rate, and is similar to classes such as airport, freeway, overpass, and intersection. The classes dense residential, medium residential, and sparse residential are very similar, and they all have relatively high error rates. It is natural that the model struggles with these similar classes, as images from similar classes will be similar in many ways. The distribution of images from similar classes likely have much overlap. In Figure 5, we show concrete examples of images from the test set that the model incorrectly classified where the incorrect predicted class is very similar to the correct class.

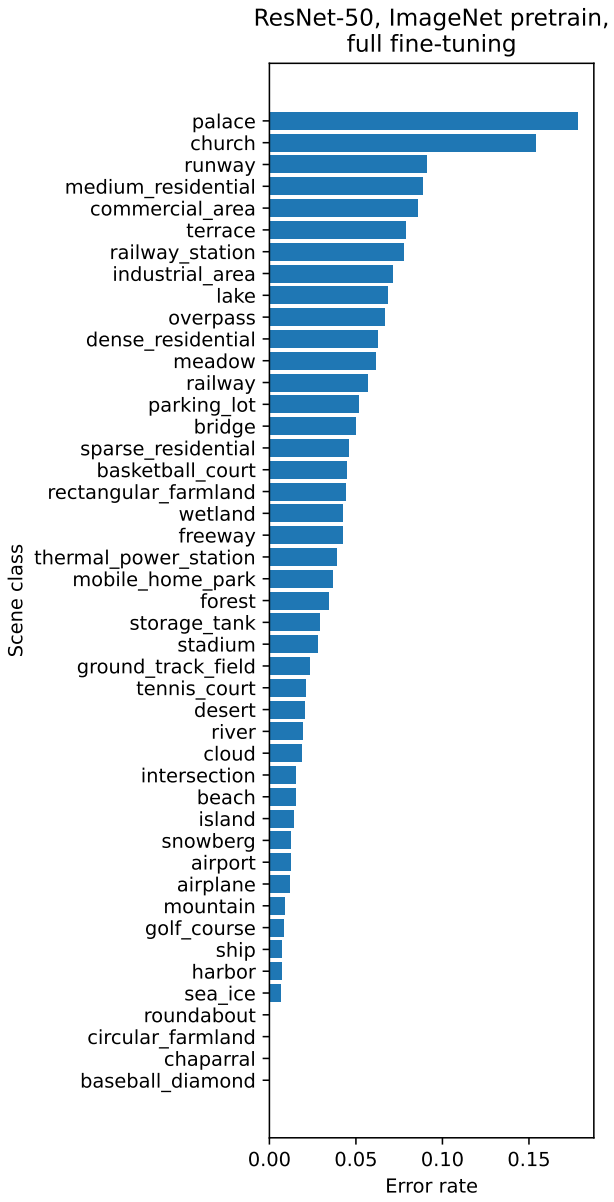

Figure 4. Per-class test set error rates for the top-performing ResNet 50 model (ImageNet pre-trained, transfer learning, full fine-tuning). The classes with the highest error rates are those that have other similar classes in the dataset, e.g., palace and church.

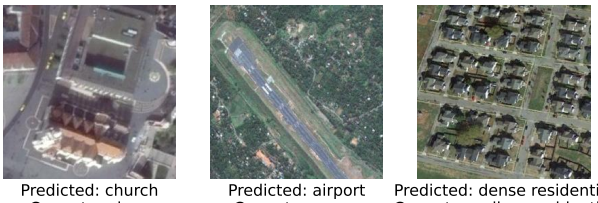

Figure 5. Example test set images that the top-performing ResNet-50 model misclassified (ImageNet pre-trained, transfer learning, full fine-tuning). In this examples, the incorrect predicted class is very similar to the correct class.

6. Conclusion

In this project, we applied transfer learning using pre-trained ResNet-50 and ViT-B/16 models for identifying land use and land cover from satellite images. After fine-tuning on the NWPU-RESISC45 dataset, we achieved 95.9% classification accuracy on the unseen test set. For our setting, transfer learning with full fine-tuning outperforms transfer learning with feature extraction only. We hypothesize that the pre-trained weights may not extract optimal features from NWPU-RESISC45, so allowing the full weights to be updated allows for more flexibility to learn altered features that are more useful specifically for NWPU-RESISC45. Surprisingly, we found that transfer learning to NWPU-RESISC45 performed better using ImageNet pre-trained weights than Sentinel-2 satellite imagery pre-trained weights.

Future work may explore additional pre-training methods to learn a model that transfers well to downstream satellite imagery tasks. For example, future work could investigate if training for more epochs, on more data, on labeled data, or with different self-supervised/unsupervised learning methods yields pre-trained models with better features for subsequent transfer learning. In addition, future work may explore learning algorithms to achieve high accuracy on all classes, including difficult classes that are highly similar to other classes in the dataset. Finally, future work may explore using multimodal language/vision models for satellite imagery. Such models could provide more detailed language descriptions of the land use and land cover in a satellite image, beyond just a label from a fixed set of classes.

Contributions

Chenchen Gu performed all the work for this project. This project is not being shared with another class.

References

[1] J. L. Ba, J. R. Kiros, and G. E. Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

[2] A. Banquet, P. Delbouve, M. Daams, and P. Veneri. Monitoring land use in cities using satellite imagery and deep learning. (28), 2022.

[3] F. Bastani, P. Wolters, R. Gupta, J. Ferdinando, and A. Kembhavi. Satlaspretrain: A large-scale dataset for remote sensing image understanding. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 16772–16782, 2023.

[4] I. Bello, W. Fedus, X. Du, E. D. Cubuk, A. Srinivas, T.-Y. Lin, J. Shlens, and B. Zoph. Revisiting resnets: Improved training and scaling strategies. *Advances in Neural Information Processing Systems*, 34:22614–22627, 2021.

[5] G. Cheng, J. Han, and X. Lu. Remote sensing image scene classification: Benchmark and state of the art. *Proceedings of the IEEE*, 105(10):1865–1883, 2017.

[6] G. Christie, N. Fendley, J. Wilson, and R. Mukherjee. Functional map of the world. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6172–6180, 2018.

[7] J. Donahue, Y. Jia, O. Vinyals, J. Hoffman, N. Zhang, E. Tzeng, and T. Darrell. Decaf: A deep convolutional activation feature for generic visual recognition. In *International conference on machine learning*, pages 647–655. PMLR, 2014.

[8] 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.

[9] European Space Agency. Sentinel-2 mission. `https://sentiwiki.copernicus.eu/web/s2-mission`.

[10] X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256. JMLR Workshop and Conference Proceedings, 2010.

[11] K. He, H. Fan, Y. Wu, S. Xie, and R. Girshick. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9729–9738, 2020.

[12] K. He, X. Zhang, S. Ren, and J. Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034, 2015.

[13] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

[14] P. Helber, B. Bischke, A. Dengel, and D. Borth. Introducing eurosat: A novel dataset and deep learning benchmark for land use and land cover classification. In *IGARSS 2018-2018 IEEE International Geoscience and Remote Sensing Symposium*, pages 204–207. IEEE, 2018.

[15] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4700–4708, 2017.

[16] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. pmlr, 2015.

[17] N. Jean, M. Burke, M. Xie, W. M. Davis, D. B. Lobell, and S. Ermon. Combining satellite imagery and machine learning to predict poverty. *Science*, 353(6301):790–794, 2016.

[18] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In Y. Bengio and Y. LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

[19] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25, 2012.

[20] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, and B. Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 10012–10022, 2021.

[21] I. Loshchilov and F. Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

[22] T. maintainers and contributors. Torchvision: Pytorch's computer vision library. `https://github.com/pytorch/vision`, 2016.

[23] National Aeronautics and Space Administration. Landsat science. `https://landsat.gsfc.nasa.gov/`.

[24] M. Neumann, A. S. Pinto, X. Zhai, and N. Houlsby. In-domain representation learning for remote sensing. *arXiv preprint arXiv:1911.06721*, 2019.

[25] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

[26] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015.

[27] A. Sharif Razavian, H. Azizpour, J. Sullivan, and S. Carlsson. Cnn features off-the-shelf: an astounding baseline for recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pages 806–813, 2014.

[28] S. Shen, S. Seneviratne, X. Wanyan, and M. Kirley. Firerisk: A remote sensing dataset for fire risk assessment with benchmarks using supervised and self-supervised learning, 2023.

[29] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[30] G. Sumbul, M. Charfuelan, B. Demir, and V. Markl. Bigearthnet: A large-scale benchmark archive for remote sensing image understanding. In *IGARSS 2019-2019 IEEE International Geoscience and Remote Sensing Symposium*, pages 5901–5904. IEEE, 2019.

[31] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.

[32] H. Touvron, M. Cord, M. Douze, F. Massa, A. Sablayrolles, and H. Jégou. Training data-efficient image transformers & distillation through attention. In *International conference on machine learning*, pages 10347–10357. PMLR, 2021.

[33] G. Tseng, I. Zvonkov, C. L. Nakalembe, and H. Kerner. Cropharvest: A global dataset for crop-type classification. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*, 2021.

[34] C. J. Tucker and J. R. G. Townshend. Strategies for monitoring tropical deforestation using satellite data. *International Journal of Remote Sensing*, 21(6-7):1461–1471, 2000.

[35] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[36] V. Vryniotis. How to train state-of-the-art models using torchvision's latest primitives. https://pytorch.org/blog/how-to-train-state-of-the-art-models-using-torchvision-latest-primitives/, 2021.

[37] Y. Wang, N. A. A. Braham, Z. Xiong, C. Liu, C. M. Albrecht, and X. X. Zhu. Ssl4eo-s12: A large-scale multimodal, multitemporal dataset for self-supervised learning in earth observation [software and data sets]. *IEEE Geoscience and Remote Sensing Magazine*, 11(3):98–106, 2023.

[38] Y. Yang and S. Newsam. Bag-of-visual-words and spatial extensions for land-use classification. In *Proceedings of the 18th SIGSPATIAL international conference on advances in geographic information systems*, pages 270–279, 2010.

[39] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson. How transferable are features in deep neural networks? *Advances in neural information processing systems*, 27, 2014.

[40] F. Zhuang, Z. Qi, K. Duan, D. Xi, Y. Zhu, H. Zhu, H. Xiong, and Q. He. A comprehensive survey on transfer learning. *Proceedings of the IEEE*, 109(1):43–76, 2020.