Transfer Learning

Transfer Learning

A new image classification problem

In this presentation, we will address an image classification problem using a dataset of four different types of flowers. Our goal is to build a model that can accurately classify these flowers.
We have learned how to construct simple CNN architectures, which allows us to start building a basic CNN with several convolutional layers and train the model effectively.
Additionally, we have studied popular models such as ResNet50 and VGG16, and we have covered how to define and train these models from scratch. This presents another viable approach to solving our classification problem.

Transfer Learning

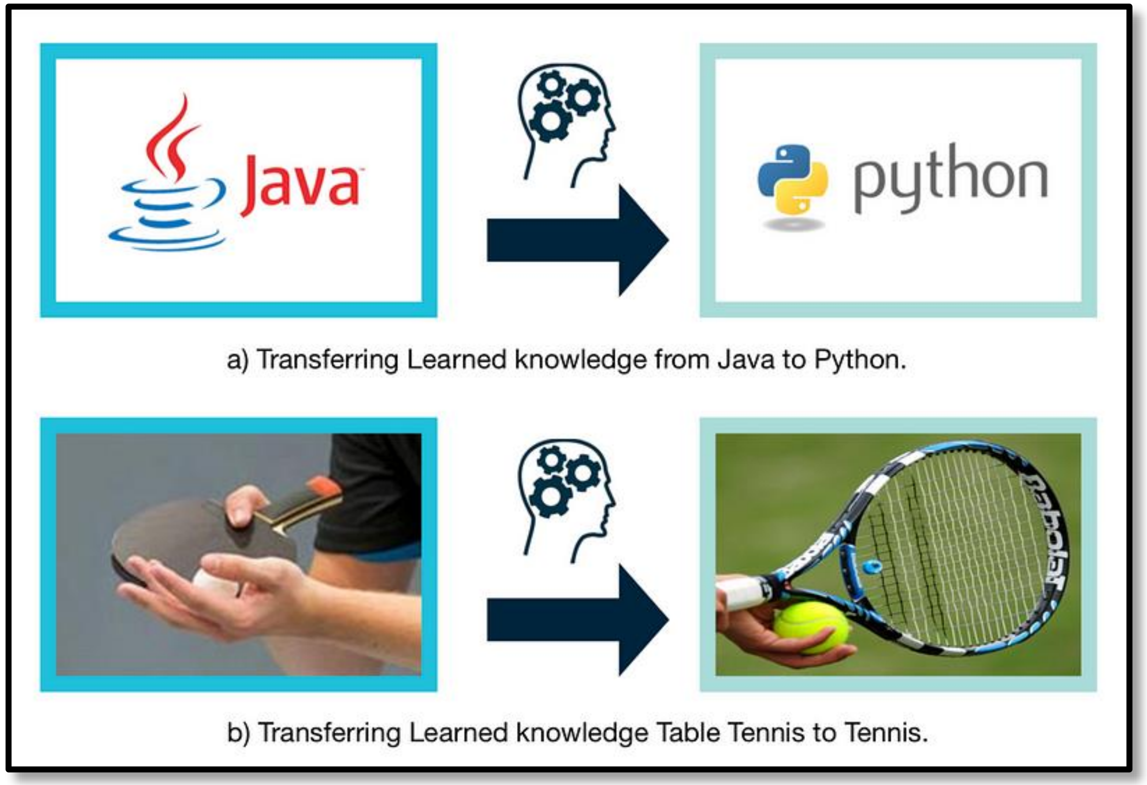
The limitations of our solutions

Building a simple model works effectively for straightforward datasets like MNIST and many others. However, when our images contain complexities, these models may struggle to extract the intricate features from the images.
Constructing larger and deeper models, such as ResNet50, will undoubtedly help us capture the complex features present in the images. However, training these models can be very time-consuming. Additionally, using these models may lead to overfitting, especially if our dataset is limited, as larger models are more prone to overfitting due to their complexity.

Transfer Learning

How Humans learn new things?

Imagine you have been playing table tennis for five years and are now part of a professional team. One weekend, your friend invites you to play tennis with him.
Since you have never played tennis before, how long do you think it would take you to learn? You would likely reach a level where you can enjoy playing tennis in less than an hour.
Having played a similar sport, table tennis, you can transfer what you have learned to this new sport and use your previous knowledge to effectively learn this new skill.
We will apply the same technique in this presentation to address the image classification problem.


Java
python
a) Transferring Learned knowledge from Java to Python.
b) Transferring Learned knowledge Table Tennis to Tennis.

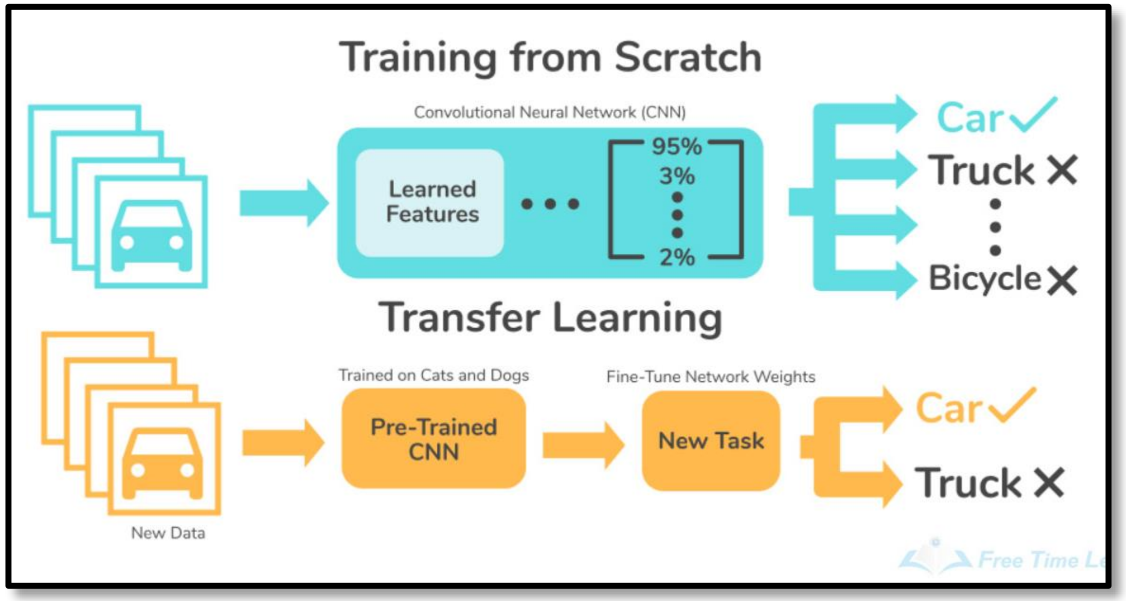
Source: www.medium.com

Transfer Learning

Transfer Learning

Similar to the tennis example, we can utilize a pretrained model to address a new problem. This approach allows us to leverage the knowledge that the model has gained from its previous training. The process of utilizing and adjusting a pretrained model to solve a new problem is known as transfer learning.

We have observed that different layers of a CNN extract different features. The early layers typically capture basic features, such as edges and corners. These features are generally useful regardless of the new dataset, so there is no need to retrain these layers. This is a key concept in transfer learning.



Source: www.skyengine.ai

Transfer Learning

Similar features in different datasets

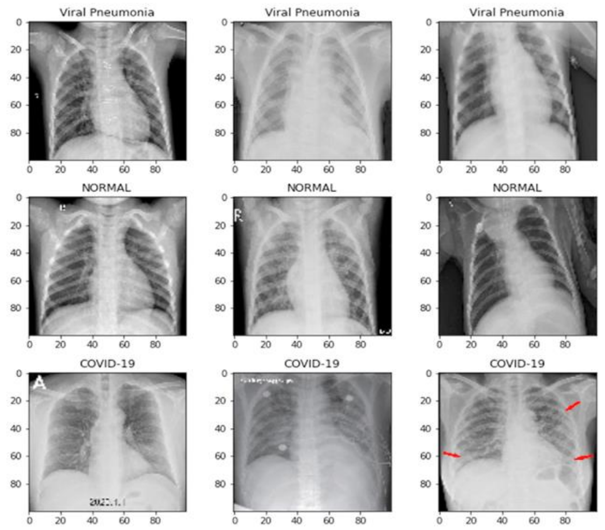
Sometimes the new dataset we are using is quite similar to the dataset on which the model was previously trained.
For example, imagine we have trained a model to classify cats and dogs. Now, we have a dataset with different breeds of cats. This new dataset is very similar to the previous one, as many features—such as the tail, ears, eyes, fur, and other characteristics—are comparable in both datasets. Therefore, we expect that the knowledge the model gained from its previous training will significantly aid its performance on the new task.

Source: www.kaggle.com

Transfer Learning

Similar features in different datasets

At times, the two datasets may not resemble each other significantly. For example, consider ImageNet and a dataset of medical images. At first glance, these two datasets appear to have little in common. However, most of the basic features extracted in the initial layers of the CNN are likely to be the same.
These layers typically capture features such as corners, edges, and other fundamental characteristics. Therefore, even in cases involving dissimilar datasets, transfer learning can still be beneficial.



Source: www.kaggle.com

Transfer Learning

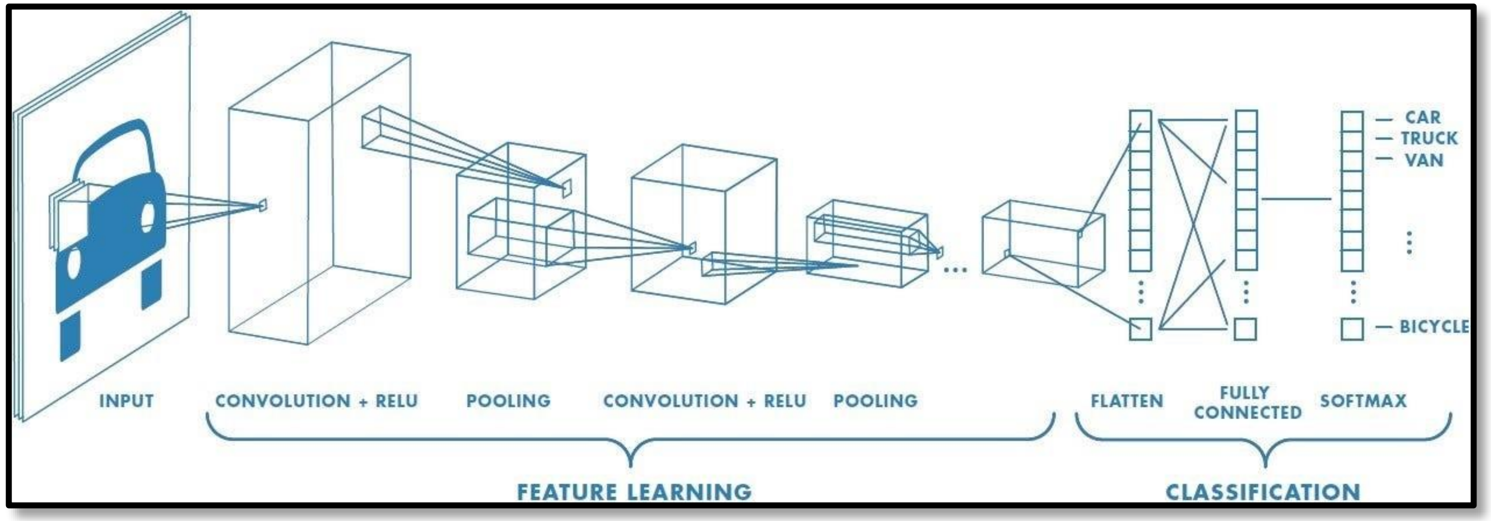
Different types of transfer learning

The key to successful transfer learning is finding the best way to adjust the pretrained model. This is a subtle process that requires a profound understanding of both the new dataset and the dataset on which the model was previously trained.
There are three different methods for solving a new problem. As designers of convolutional neural networks, we should master the art of choosing the right method to address our limitations. In the next slides, we will discuss each method and explore how to choose among our options.

Transfer Learning

1.Training from scratch

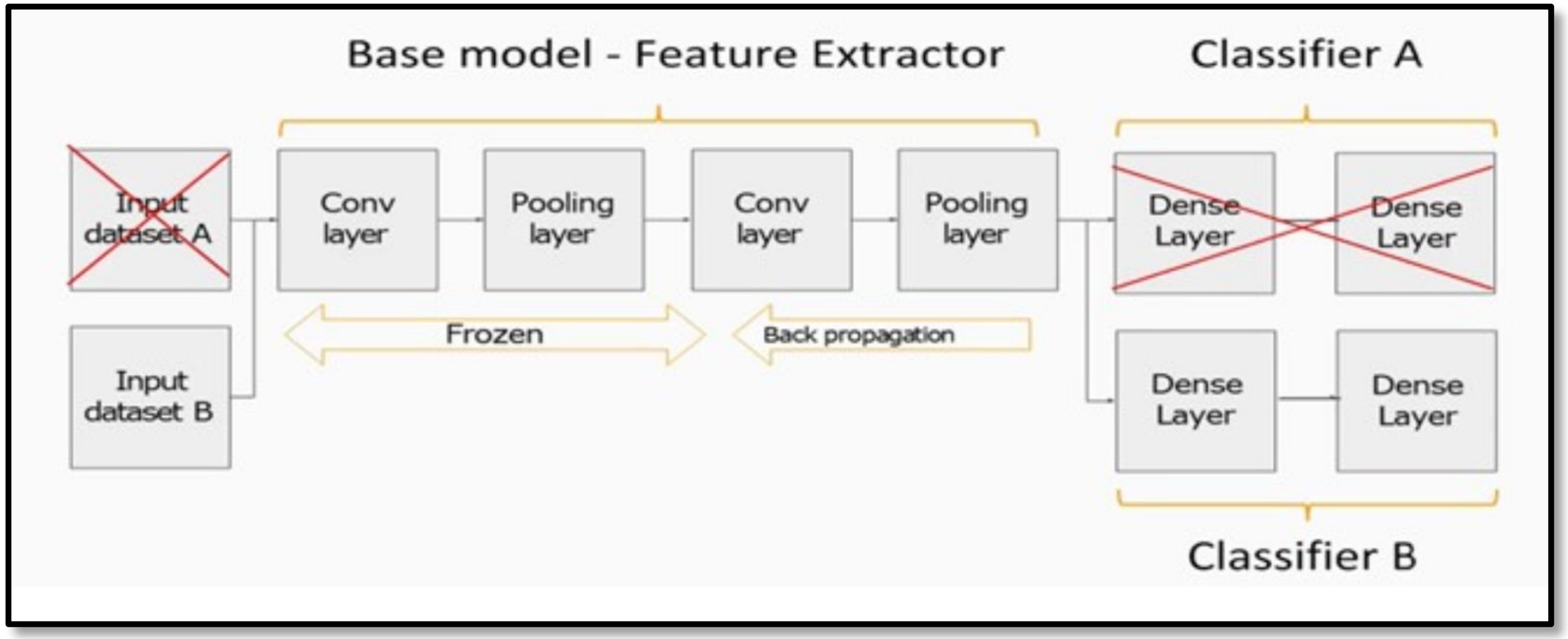
We should always remember the initial problem we were trying to solve. We began exploring transfer learning for two main reasons: either we did not have enough time or resources to train a model from scratch, or our new dataset was not large enough to prevent overfitting.
If neither of these issues affects our new image classification task, we can simply train our model from scratch.



Transfer Learning

2.Fine Tuning

When our new dataset is large and similar to the dataset on which the model was trained, we can utilize fine-tuning.
In this method, we freeze the first few layers of the base model and modify the fully connected layers at the end of the network.
We train the new layers and fine-tune the deeper layers of the base model on the new dataset, continuing the training until convergence.

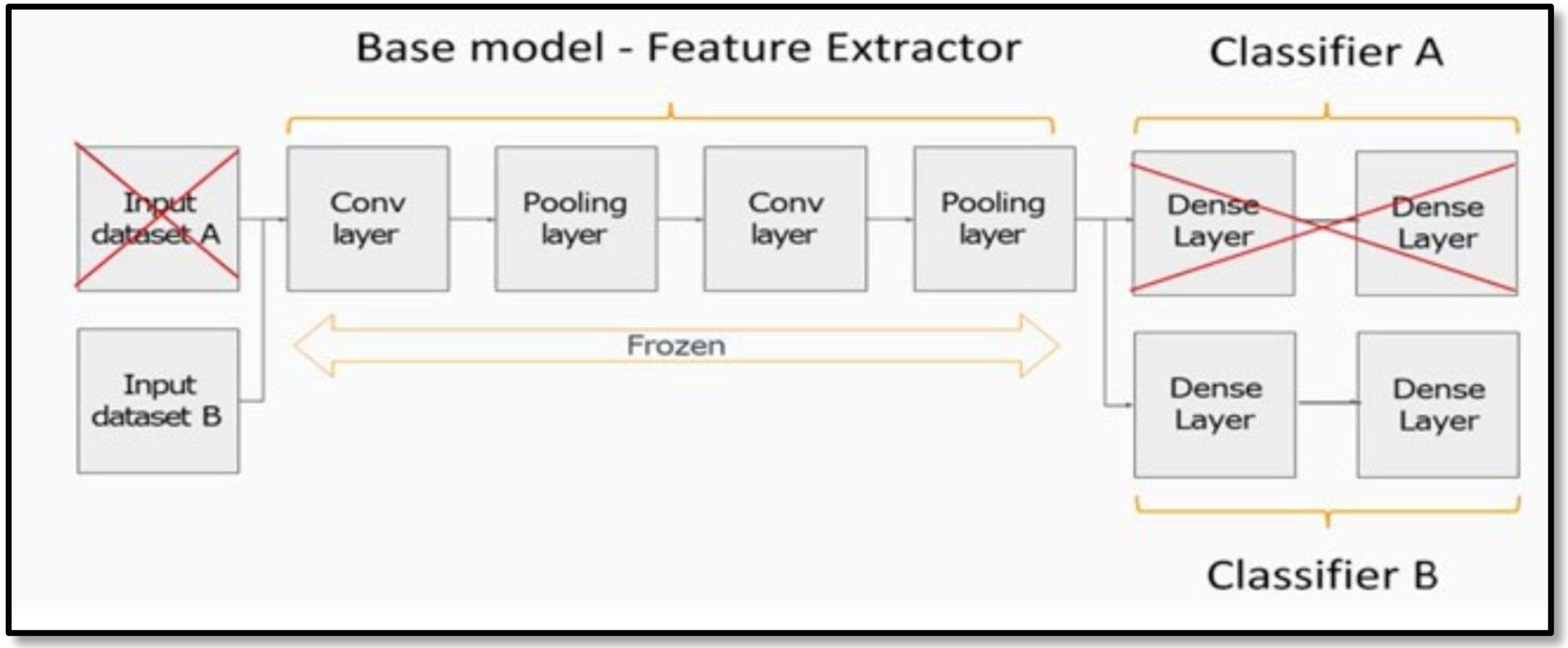


Source: Hands-On Transfer Learning with TensorFlow 2.0, Margaret Maynard

Transfer Learning

3.Changing the classifiers

When our new dataset is small but similar to the dataset on which the model was trained, we can modify the classifiers.
In this method, we freeze all the layers of the base model and then add a classifier consisting of fully connected layers on top.
Since our dataset is small, utilizing the fine-tuning method may result in overfitting.



Source: Hands-On Transfer Learning with TensorFlow 2.0, Margaret Maynard

Transfer Learning

Back to the example

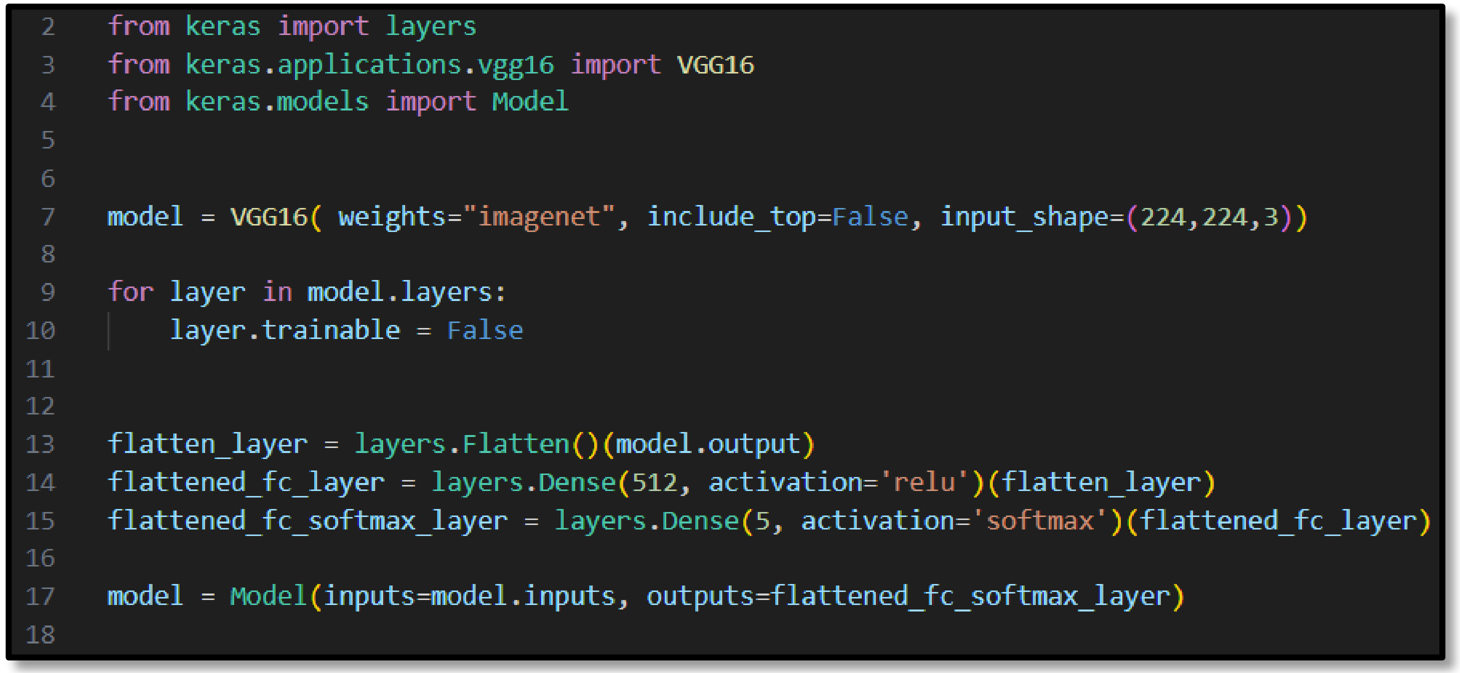
Let us now explore the flower classification example. We are going to solve this problem using transfer learning. We need to decide which CNN architecture will be utilized. For this example, we will use a VGG16 model, which is pre-trained on the ImageNet dataset.
Now, we have to determine how we are going to perform transfer learning. We need to compare the new dataset with ImageNet. ImageNet has classes related to plants and flowers, and it even includes a class named "daisy." Therefore, our dataset is similar to ImageNet. As a result, we will not fine-tune the pre-trained model, and we will only modify the classifier layers.

various classes of plants in ImageNet

Transfer Learning

Defining the model

Now, we will explore how to use transfer learning in Python, step by step. We will focus on defining the model and training it, while other steps, such as importing and preprocessing the data, are similar to what we have already covered.
First, we need to create an instance of the base model. In this case, we will create a VGG16 model that is pre-trained on the ImageNet dataset. By setting the include_top parameter to False, we will remove the base model's classifiers. Next, we will set the trainable parameters of all the layers in the base model to False to freeze them. Finally, we will add our new classifiers.



```python
from keras import layers
from keras.applications.vgg16 import VGG16
from keras.models import Model


model = VGG16( weights="imagenet", include_top=False, input_shape=(224,224,3))

for layer in model.layers:
    layer.trainable = False


flatten_layer = layers.Flatten()(model.output)
flattened_fc_layer = layers.Dense(512, activation='relu')(flatten_layer)
flattened_fc_softmax_layer = layers.Dense(5, activation='softmax')(flattened_fc_layer)

model = Model(inputs=model.inputs, outputs=flattened_fc_softmax_layer)

```

Transfer Learning

Training the model

Now that we have successfully defined the model, we can begin the training process. This process is similar to training any other model.
First, we need to import and preprocess the data. Next, we will split the data into training and testing sets. We will use the training dataset to train the model. It is important to remember that the parameters in the frozen layers will not change during training. In our example, the fully connected layers that we added on top of the base model will be optimized during training, while the other parameters remain frozen. We will evaluate and utilize our model once it converges.

Summary

In this lecture, a new approach to solving image classification problems was introduced. First, we analyzed the limitations of other approaches and how transfer learning can address them.
Next, different methods of transfer learning were introduced. We covered how to determine which method to use based on the characteristics of our new dataset and the dataset that the base model was initially trained on.
Finally, we defined a pre-trained model and modified its classifiers to better understand the process of transfer learning in Python and TensorFlow.