Image Classification with Transfer Learning and FastAI

Ujwal Gullapalli[1], Lei Chen[1(✉)], and Jinbo Xiong[2]

[1] Georgia Southern University, Statesboro, GA 30460, USA
LChen@georgiasouthern.edu
[2] Fujian Normal University, Fuzhou 350117, China

Abstract. Today deep learning has provided us with endless possibilities for solving problems in many domains. Diagnosing diseases, speech recognition, image classification, and targeted advertising are a few of its applications. Starting this process from scratch requires using large amounts of labeled data and significant cloud processing usage. Transfer learning is a deep learning technique that solves this problem by making use of a model that is pre-trained for a certain task and using it on a different task of a related problem. Therefore, the goal of the project is to utilize transfer learning and achieve near-perfect results using a limited amount of data and computation power. To demonstrate, an image classifier using FastAI that detects three types of birds with up to 94% accuracy is implemented. This approach can be applied to solve tasks that are limited by labeled data and would gain by knowledge learned from a related task.

Keywords: Transfer learning · Deep learning · Image classification

1 Introduction

Traditionally, deep learning models are used in isolation. To solve a given task, they are trained on millions of data points. They achieve good results for that particular task, but when a different task needs to be solved, the model would again need to be trained on a large dataset that is related to that specific problem. The previous model cannot be used to solve the new task as the model will be biased from its training data and it will not fit well. The critical issue is that most models to solve complex tasks would require a large amount of labeled data and computing power. Transfer learning can solve these issues by making use of the knowledge gained on previously learned tasks and applying it to new tasks. The transfer learning model will work effectively if the pre-trained model is well-generalized. For example, the state-of-the-art image classification models such as VGG-16, VGG-19, InceptionV3, XCeption, and Resnet50 are trained on the ImageNet database [1]. The database contains over 1.2 million images. These models are generalized and their accuracy is determined on a specific dataset.

J. Xiong et al. (Eds.): MobiMedia 2021, LNICST 394, pp. 796–806, 2021.
https://doi.org/10.1007/978-3-030-89814-4_59

When these pre-trained models are directly applied to a different task or a similar problem they will still suffer a significant loss in performance. Therefore, the primary motivation for the paper was to explore transfer learning by building an image classifier that classifies three types of birds with less data. The procedure and techniques used for building the classifier is inspired from Jeremy Howard's and Sylvain's work on Deep Learning [2]. The development environment for implementing transfer learning can be built locally in an IDE such as Pycharm, Spyder, Visual Studio, or by using a cloud service provider such as Google Colab, PaperSpace, AWS. In this case, Paperspace cloud service is used. It provides free and paid computer resources and uses Jupyter Notebooks as its development environment. To implement the model, FastAI, a deep learning library that runs on top of PyTorch is used. It is a library designed to be quick and easy to rapidly deploy state-of-the-art models to solve problems [3].

2 Related Works

Transfer learning has been around for a few decades, but it has received less interest compared to other areas of machine learning such as unsupervised and reinforcement learning. Andrew Ng, a renowned computer scientist, and a pioneer in AI, believes that apart from supervised learning, transfer learning will be the driver of commercial success for machine learning [4]. Transfer learning is being used across multiple domains. One of the key areas is learning from one language and applying that knowledge to another language. There has been interesting research in this area like zero-shot translation [5]. It uses a single Neural Machine Translation model to translate between multiple languages and has achieved state-of-the-art results.

The other interesting area being utilized is for simulation learning. The model is trained in simulation and the knowledge is transferred to the real-world robot. This is because the robots can be slow and expensive to train [6]. Similarly, self-driving technology is also utilizing simulation training and there are open-source resources where a self-driving car can be trained [7,8]. When it comes to image classification most of the pre-trained models are based on convolutional neural networks that are trained on the Imagenet database. [9] studied if the models that performed better on the Imagenet database also performed well on other vision tasks. They found when the models are used as feature extractors or for finetuning, there was a strong correlation between Imagenet accuracy and transfer accuracy. [10] looked at how transferable are the features in deep neural networks. One of their observation was that the performance benefits of transferring features decrease the more dissimilar source task and target task are.

Some of the challenges in transfer learning are figuring out the best features and measuring the transferability of a model. [11] proposed attentive feature distillation and selection (AFDS) technique adjusts the strength of transfer learning regularization and figures out the important features to transfer. They deployed AFDS on Resnet101 and were able to outperform all existing transfer learning

methods. [12] proposed a new metric called LEEP (Log Expected Empirical Prediction) that can predict the performance and convergence speed of transfer and meta-transfer learning methods. It outperformed other metrics such as negative conditional entropy and H scores.

3 Methodology

To demonstrate transfer learning, we built an image classifier that classifies three different types of birds. We used a model pre-trained on an Imagenet and applied it to our task which is to classify three different types of birds. We used a process knows as fine-tuning, that mainly comprised of three steps:

- **Importing the Model:** A model that is trained on a benchmark dataset like Imagenet is imported. It can classify everyday objects, animals, birds, etc. The last layer of the pre-trained model that predicts various classes is removed and replaced with the output layer consisting of three classes of birds.
- **Freezing:** After modifying the output layer, the layers that have come from the pre-trained model were frozen. This means that frozen layer weights will not be updated when the model is trained. Only the modified layer is updated during training.
- **Training:** In final step, the model is trained till it achieves a lowest error rate without overfitting.

There are many types of transfer learning techniques. In some of the cases, the last few layers are removed instead of just the head. This is because, the pre-trained model initial layers can act as general feature extractors and last layers will be more geared towards the original dataset.

4 Building the Bird Classifier

The birds we are going to classify are three birds that are commonly found in the backyard in the state of Georgia. These are Northern Cardinal, Blue Jay, and Yellow-Rumped Warbler (Fig. 1). The image classifier consists of a GUI within the Jupyter notebook. It takes bird image as input and predicts its class.

4.1 Setup Environment

Importing required libraries: FastAI to implement the model and JMD image scraping library to get our data. The development was done in Jupyter Notebook environment.

Fig. 1. Types of birds to be classified. (Color figure online)

```
import fastbook
from fastbook import *
fastbook.setup_book()
from fastai.vision.widgets import *

from jmd_imagescraper.core import *
from pathlib import Path
from jmd_imagescraper.imagecleaner import *
```

4.2 Data Collection

Created the directories to store data.

```
bird=['Northern Cardinal','Blue Jay','Yellow Rumped warbler']
path = Path().cwd()/"bird"
```

The problem required images of three different birds: Blue Jay, Northern Cardinal, and Yellow-Rumped Warbler. A total of 150 images per bird type are collected from the DuckDuckGo search engine using the JMD image scraper (Fig. 2).

```
duckduckgo_search(path,"Northern Cardinal",
"Northern Cardinal",max_results=150)

duckduckgo_search(path,"Blue Jay",
"Blue Jay",max_results=150)

duckduckgo_search(path,"Yellow Rumped Warbler"
,"Yellow rumped warbler",max_results=150)
```

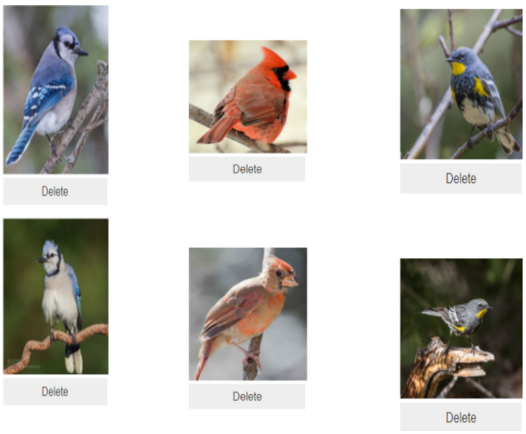

Fig. 2. Collected images from DuckDuckGo search engine. (Color figure online)

4.3 Model Creation

FastAI library provides a simple Datablock method to feed data into our model for training.

```
bird=DataBlock(
    blocks=(ImageBlock,CategoryBlock),
    get_items=get_image_files,
    splitter=RandomSplitter(valid_pct=0.2,seed=42),
    get_y=parent_label,
    item_tfms=Resize(128)
   )
```

The parameters used are as follows:

- blocks: ImageBlock is the input data that is used for the model and category block refers to the labels.
- get_items: get_image_files is used to get the image file names from the path.
- splitter: It is used to specify the method to split the validation set and the training set. Here, the data is split randomly and 20% is allocated to validation set. A seed is used to make sure that every time the program runs, the validation set stays the same.
- get_y: It is specified to locate labels of the data. The method parent_label is going to look at name of the parent in the path of the image file.
- item_tfms: The data from the internet can be of any size. Hence, image is resized into 128×128.

After creating the datablock, a dataloader is created. A dataloader is used to grab a bunch of images as a batch for GPU processing. This makes the GPU process much faster.

```
dls=bird.dataloaders(path)
```

Checking the images from the validation batch (Fig. 3).

```
dls.valid.show_batch(max_n=4,nrows=1)
```

Fig. 3. Images from validation batch.

4.4 Data Transformation

The default resizing method crops the image to fit a square. This might lead to losing some of the information from the picture. There are other resize methods such as squishing, stretching, padding, etc. Squishing or stretching the image can lead to uneven shapes of the bird. This can affect the performance of the model. After trying different image transformation methods, the RandomResizedCrop method gave the best performance for this problem. It crops the random size of the original image and a random aspect ratio of the original aspect ratio is created. This also helps make the model less likely to overfit as it sees a different part of the image every time (Fig. 4).

```
bird = bird.new(
item_tfms=RandomResizedCrop(224, min_scale=0.4),
batch_tfms=aug_transforms())

dls = bird.dataloaders(path)

dls.train.show_batch(max_n=4, nrows=1, unique= True)
```

Fig. 4. Training set images after transformation.

4.5 Model Training

There are a lot of architectures such as VGG19, InceptionNET, and ResNet that can be used for this image classification problem. Here, a Convolution Neural Network using the Resnet architecture consisting of 18 layers is created. This model is pre-trained on Imagenet database and it can classify general images into 1000 classes. The metric used was error rate, which tells how frequently the model is making incorrect predictions.

```
learn=cnn_learner(dls, resnet18, metrics=error_rate)
```

The model is then trained for 4 epochs. An epoch specifies the number of times the image will be used as training data.

```
learn.fine_tune(4)
```

epoch	train_loss	valid_loss	error_rate	time
0	1.300787	0.381271	0.177778	00:04

epoch	train_loss	valid_loss	error_rate	time
0	0.208502	0.319453	0.144444	00:05
1	0.132655	0.240997	0.088889	00:05
2	0.087438	0.250987	0.077778	00:05
3	0.070439	0.332649	0.133333	00:05

Fig. 5. Training results for each epoch.

The results (Fig. 5) showed that after training for just 3 epochs the error rate was down to around 7% from 17%. This is for the images randomly downloaded from the internet. There is a chance that there might be false data and this can affect the model performance. FastAI provides a way to check the images the model is having trouble predicting. Checking the images with top 5 loses (Fig. 6).

```
interp.plot_top_losses(5, nrows=5)
```

The top 5 loses showed that the model is having trouble predicting the Yellow-Rumped Warbler. The first image in the top loss results is blurry and the second image is not relevant and they should be discarded from the dataset.

The data was cleaned with FastAI inbuilt cleaner widget (Fig. 7).

```
cleaner = ImageClassifierCleaner(learn)
cleaner
```

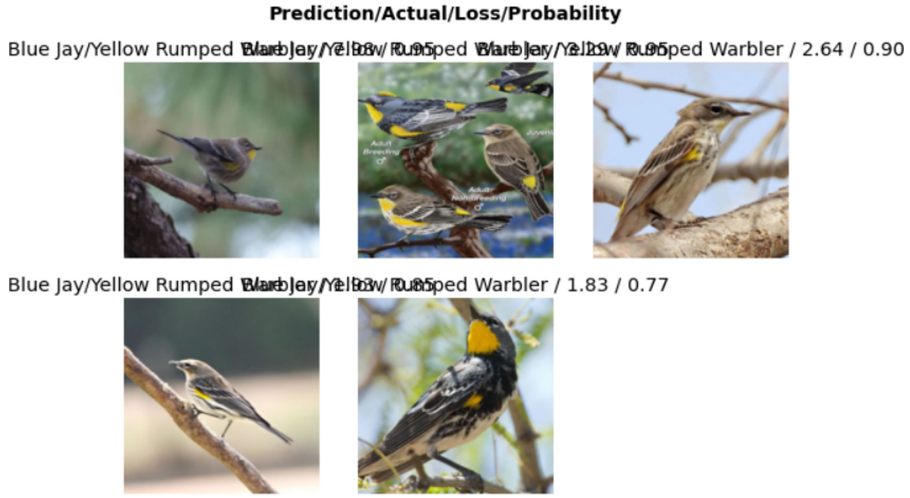

Fig. 6. Top 5 images the model is having trouble classifying. (Color figure online)

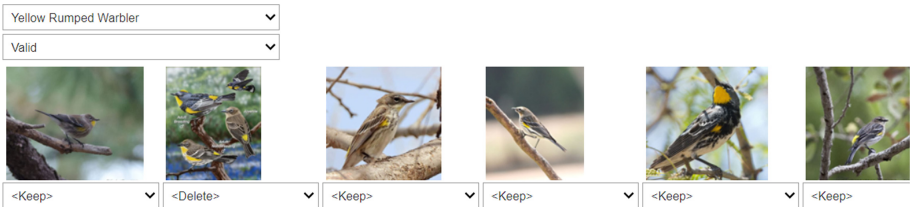

Fig. 7. FastAI inbuilt cleaner.

After cleaning the data and retraining for 2 epochs. The error rate was reduced to around 5% (Fig. 8).

epoch	train_loss	valid_loss	error_rate	time
0	0.180719	0.165416	0.057471	00:05
1	0.121468	0.165792	0.057471	00:05

Fig. 8. Retraining results

4.6 Results

The results were visualized using a confusion matrix (Fig. 9).

```
interp = ClassificationInterpretation.from_learner(
learn)
interp.plot_confusion_matrix()
```

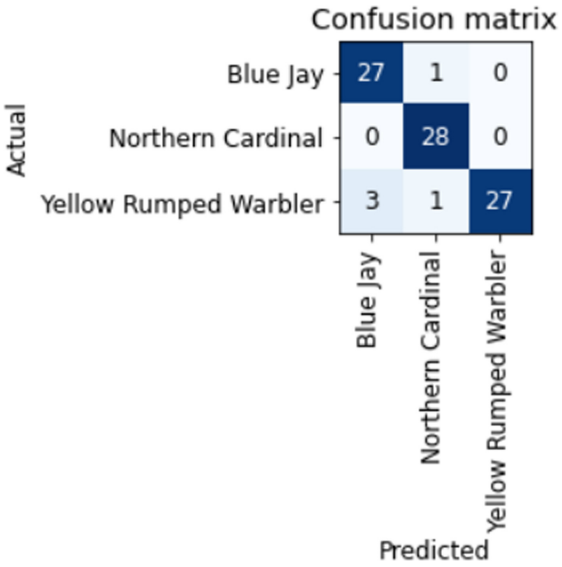

Fig. 9. Results visualized using confusion matrix. (Color figure online)

The confusion matrix showed that there were four Warblers misclassified as Blue Jays and Northern Cardinal and one Blue Jay was misclassified as Northern cardinal. We have a total of 5 incorrect predictions out of 87 making our error rate 0.057%.

4.7 Building a Classifier Widget

In the final step, a classifier widget within Jupyter notebook environment was built. It looks similar to an image upload GUI. After building the widget, the classifier was tested using random images from Google image search. The first step is to export the model, so that it can be reused for other tasks or it can be deployed in an application.

Here, the model is exported and loaded again to build the classifier widget.

```
learn.export()
learn_export = load_learner('export.pkl')
```

A function is defined to build the classifier widget.

```
btn_upload = widgets.FileUpload()
btn_run = widgets.Button(description='Classify')
out_pl = widgets.Output()
lbl_pred = widgets.Label()

def on_click_classify(change):
```

```
    img = PILImage.create(btn_upload.data[-1])
    out_pl.clear_output()
    with out_pl: display(img.to_thumb(128,128))
    pred,pred_idx,probs = learn_export.predict(img)
    lbl_pred.value = f'Prediction: {pred}; Probability: {probs[
                                        pred_idx]:.04f}'

btn_run.on_click(on_click_classify)
```

The widget is loaded.

```
VBox([widgets.Label('Upload Bird!'),
      btn_upload, btn_run, out_pl, lbl_pred]
```

A random image of a warbler from google image search is uploaded and tested.

The classifier predicted that the image is of a Yellow-Rumped Warbler with 99% probability (Fig. 10).

5 Summary

Based on the results, it can be summarized as the following:

- A large labeled dataset is not required to solve a task using transfer learning.
- The time taken to train the model is very little, which also reduces the computation costs.
- FastAI Library is very effective in implementing state-of-the-art models and transfer learning.

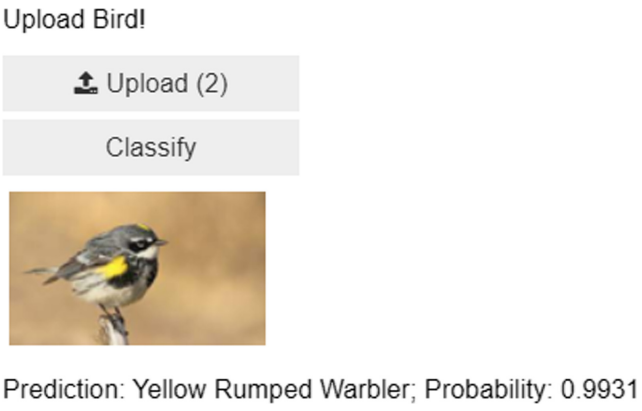

Fig. 10. Testing a random warbler image from Google. (Color figure online)

There are many techniques of transfer learning. The fine-tuning technique demonstrated here takes a pre-trained model for one given task and then tunes the model to make it perform on a similar task. This simple process can help us solve real-world problems. For example, we can build an Image detection system that tracks endangered birds or a warning system that detects a poisonous jellyfish in the water.

6 Conclusion

This research explored transfer learning and its importance. It also applied the transfer learning technique using a Resnet18 model for bird classification and achieved great results. This was done only using 450 images and it took less than one minute to train the model. Therefore, by leveraging the power of transfer learning many tasks that do not have large amounts of labeled data can possibly be solved, which is typically the constraint in a real-world setting.

References

1. Image Net Database. Image Net. http://www.image-net.org/. Accessed Apr 2021
2. Howard, J., Gugger, S.: Deep Learning for Coders with Fastai and PyTorch: AI Applications Without a PhD. Information, 1st edn., July 2020. https://www.oreilly.com/library/view/deep-learning-for/9781492045519/
3. Howard, J., Gugger, S.: Fastai: a layered API for deep learning. Information **11**(2), 108 (2020). https://doi.org/10.3390/info11020108
4. Andrew, N.G.: Deep Learning for Building AI Systems (2016). https://nips.cc/Conferences/2016/TutorialsIntros
5. Johnson, M., et al.: Google's multilingual neural machine translation system: enabling zero-shot translation. TACL **5**, 339–351 (2017). https://doi.org/10.1162/tacl_a_00065
6. Rusu, A.A., Vecerik, M., Rothörl, T., Heess, N., Pascanu, R., Hadsell, R.: Sim-to-Real Robot Learning from Pixels with Progressive Nets (2016). arXiv:1610.04286
7. OpenAI, Universe. Open AI Universe. https://github.com/openai/universe. Accessed Apr 2021
8. Udacity. Self-Driving-Car-Sim. Udacity. https://github.com/udacity/self-driving-car-sim. Accessed Apr 2021
9. Kornblith, S., Shlens, J., Le, Q.V.: Do better ImageNet models transfer better? In: 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). IEEE (2019)
10. Yosinski, J., Clune, J., Bengio, Y., Lipson, H.: How transferable are features in deep neural networks? In: Advances in Neural Information Processing Systems 27, pp. 3320–3328, December 2014. arXiv:1411.1792
11. Wang, K., Gao, X., Zhao, Y., Li, X., Dou, D., Xu, C.: Pay Attention to Features, Transfer Learn Faster CNNs (2019). openreview.net. https://openreview.net/forum?id=ryxyCeHtPB
12. Nguyen, C.V., Hassner, T., Seeger, M., Archambeau, C.: LEEP: A New Measure to Evaluate Transferability of Learned Representations (2020). arXiv:2002.12462