

f (<https://www.facebook.com/AnalyticsVidhya>) | **t** (<https://twitter.com/analyticsvidhya>)

g+ (<https://plus.google.com/+Analyticsvidhya/posts>)

in (<https://www.linkedin.com/groups/Analytics-Vidhya-Learn-everything-about-5057165>)

Home (<https://www.analyticsvidhya.com/>) | Blog (<https://www.analyticsvidhya.com/blog/>)

Jobs (<https://www.analyticsvidhya.com/jobs/>) | Trainings (<https://www.analyticsvidhya.com/trainings/>)

Learning Paths (<https://www.analyticsvidhya.com/learning-paths-data-science-business-analytics-business-intelligence-big-data/>)

Discuss (<https://discuss.analyticsvidhya.com>)

Datahack Summit 2017 (<https://www.analyticsvidhya.com/datahacksummit/>)



(<https://www.analyticsvidhya.com>)



(<https://www.analyticsvidhya.com>)

[/datahacksummit/?utm_source=avblog_header&utm_medium=web&utm_campaign=banner](https://www.analyticsvidhya.com/datahacksummit/?utm_source=avblog_header&utm_medium=web&utm_campaign=banner))

HOME (<https://www.analyticsvidhya.com/>) LEARN ▾ ENGAGE ▾ COMPETE ▾ GET HIRED ▾



DATAHACK SUMMIT 2017 (<https://www.analyticsvidhya.com/datahacksummit>)

[/?utm_source=av_header_menu&utm_medium=avsite](https://www.analyticsvidhya.com/datahacksummit/?utm_source=av_header_menu&utm_medium=avsite))

CONTACT US (<https://www.analyticsvidhya.com/contact/>)

WE ARE HIRING! (<https://www.analyticsvidhya.com/about-me/career-analytics-vidhya/>)

Home (<https://www.analyticsvidhya.com/>) > Deep Learning
 (<https://www.analyticsvidhya.com/blog/category/deep-learning/>) > Transfer
 learning & The art of using Pre-trained Models in Deep Learning
 (<https://www.analyticsvidhya.com/blog/2017/06/transfer-learning-the-art-of-fine-tuning-a-pre-trained-model/>)

Transfer learning & The art of using Pre-trained Models in Deep Learning

DEEP LEARNING ([HTTPS://WWW.ANALYTICSVIDHYA.COM/BLOG/CATEGORY/DEEP-LEARNING/](https://www.analyticsvidhya.com/blog/category/deep-learning/)) PYTHON ([HTTPS://WWW.ANALYTICSVIDHYA.COM/BLOG/CATEGORY/PYTHON-2/](https://www.analyticsvidhya.com/blog/category/python/)) UNCATEGORIZED ([HTTPS://WWW.ANALYTICSVIDHYA.COM/BLOG/CATEGORY/UNCATEGORIZED/](https://www.analyticsvidhya.com/blog/category/uncategorized/))

SHARE **f** (<http://www.facebook.com/sharer.php?u=https://www.analyticsvidhya.com/blog/2017/06/transfer-learning-the-art-of-fine-tuning-a-pre-trained-model/&t=Transfer%20learning%20%20The%20art%20of%20using%20Pre-trained%20Models%20in%20Deep%20Learning>) **t** (<https://twitter.com/home?status=Transfer%20learning%20%20The%20art%20of%20using%20Pre-trained%20Models%20in%20Deep%20Learning+https://www.analyticsvidhya.com/blog/2017/06/transfer-learning-the-art-of-fine-tuning-a-pre-trained-model/>) **g** (<https://plus.google.com/share?url=https://www.analyticsvidhya.com/blog/2017/06/transfer-learning-the-art-of-fine-tuning-a-pre-trained-model/>) **p** (<http://pinterest.com/pin/create/button/?url=https://www.analyticsvidhya.com/blog/2017/06/transfer-learning-the-art-of-fine-tuning-a-pre-trained-model/&media=https://s3-ap-south-1.amazonaws.com/av-blog-media/wp-content/uploads/2017/06/01083503/pexels-photo-221485-1.jpeg&description=Transfer%20learning%20%20The%20art%20of%20using%20Pre-trained%20Models%20in%20Deep%20Learning>)



(http://events.upxacademy.com/BigDataAnalytics?utm_source=BDAWeek-AV-Banner&utm_medium=Banner&utm_campaign=BDAWeek)

Introduction

Neural networks are a different breed of models compared to the supervised machine learning algorithms. Why do I say so? There are multiple reasons for that, but the most prominent is the cost of running algorithms on the hardware.

In today's world, RAM on a machine is cheap and is available in plenty. You need hundreds of GBs of RAM to run a super complex supervised machine learning problem – it can be



(https://www.analyticsvidhya.com/datahacksummit/?utm_source=avblog_topusers&utm_medium=web&utm_campaign=banner)



(http://www.greatlearning.educaat/analytics/?utm_source=avm&utm_medium=avmbanner&utm_campaign=pqpba+bda)



(https://upgrad.com/data-science?utm_source=AV&utm_medium=Display&utm_campaign=DS_AV_Banner)

yours for a little investment / rent. On the other hand, access to GPUs is not that cheap. You need access to hundred GB VRAM on GPUs – it won't be straight forward and would involve significant costs.

Now, that may change in future. But for now, it means that we have to be smarter about the way we use our resources in solving Deep Learning problems. Especially so, when we try to solve complex real life problems on areas like image and voice recognition. Once you have a few hidden layers in your model, adding another layer of hidden layer would need immense resources.

Thankfully, there is something called "Transfer Learning" which enables us to use pre-trained models from other people by making small changes. In this article, I am going to tell how we can use pre-trained models to accelerate our solutions.

Note – *This article assumes basic familiarity with Neural networks and deep learning. If you are new to deep learning, I would strongly recommend that you read the following articles first:*

1. What is deep learning and why is it getting so much attention? (<https://www.analyticsvidhya.com/blog/2014/06/deep-learning-attention/>)
2. Deep Learning vs. Machine Learning – the essential differences you need to know! (<https://www.analyticsvidhya.com/blog/2017/04/comparison-between-deep-learning-machine-learning/>)
3. 25 Must Know Terms & concepts for Beginners in Deep Learning (<https://www.analyticsvidhya.com/blog/2017/05/25-must-know-terms-concepts-for-beginners-in-deep-learning/>)

utm_term=DS_AV_Banner&
utm_content=DS_AV_Banner)



(http://praxis.ac.in/fulltime-bigdata-analytics/?utm_source=AVBanner)

POPULAR POSTS

- Essentials of Machine Learning Algorithms (with Python and R Codes) (<https://www.analyticsvidhya.com/blog/2017/09/common-machine-learning-algorithms/>)
- A Complete Tutorial to Learn Data Science with Python from Scratch (<https://www.analyticsvidhya.com/blog/2016/01/complete-tutorial-learn-data-science-python-scratch-2/>)
- 7 Types of Regression Techniques you should know! (<https://www.analyticsvidhya.com/blog/2016/01/7-types-of-regression-techniques-you-should-know/>)

4. Why are GPUs necessary for training Deep Learning models? (<https://www.analyticsvidhya.com/blog/2017/05/gpus-necessary-for-deep-learning/>)

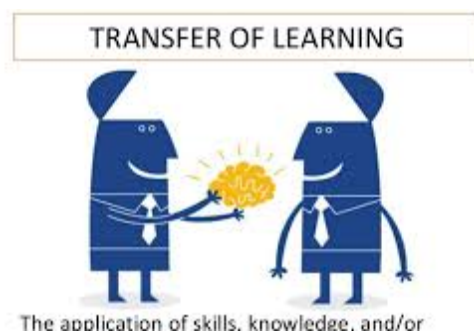
Table of Contents

1. What is transfer learning?
2. What is a Pre-trained Model?
3. Why would we use pre-trained models? – A real life example
4. How can I use pre-trained models?
 - Extract Features
 - Fine tune the model
5. Ways to fine tune your model
6. Use the pre-trained model for identifying digits
 - Retraining the output dense layers only
 - Freeze the weights of first few layers

What is transfer learning?

Let us start with developing an intuition for transfer learning. Let us understand from a simple teacher – student analogy.

A teacher has years of experience in the particular topic he/she teaches. With all this accumulated information, the lectures that students get is a concise and brief overview of the topic. So it can be seen as a “transfer” of information from the learned to a novice.



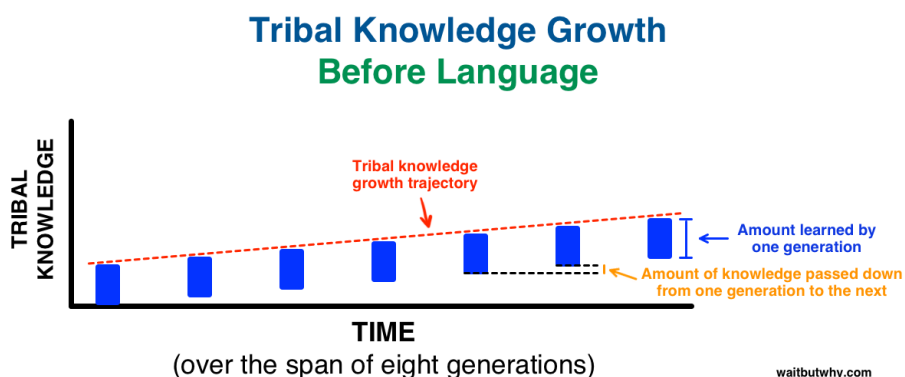
/blog/2015/08/comprehensive-guide-regression/) Understanding Support Vector Machine algorithm from examples (along with code) (<https://www.analyticsvidhya.com/blog/2017/09/understaing-support-vector-machine-example-code/>) A Complete Tutorial on Tree Based Modeling from Scratch (in R & Python) (<https://www.analyticsvidhya.com/blog/2016/04/complete-tutorial-tree-based-modeling-scratch-in-python/>) 6 Easy Steps to Learn Naive Bayes Algorithm (with codes in Python and R) (<https://www.analyticsvidhya.com/blog/2017/09/naive-bayes-explained/>) 17 Ultimate Data Science Projects To Boost Your Knowledge and Skills (& can be accessed freely) (<https://www.analyticsvidhya.com/blog/2016/10/17-ultimate-data-science-projects-to-boost-your-knowledge-and-skills/>)

attitudes that were learned in one situation to another **learning** situation (Perkins, 1992)

Keeping in mind this analogy, we compare this to neural network. A neural network is trained on a data. This network gains knowledge from this data, which is compiled as "weights" of the network. These weights can be extracted and then transferred to any other neural network. Instead of training the other neural network from scratch, we **"transfer"** the learned features.

Now, let us reflect on the importance of transfer learning by relating to our evolution. And what better way than to use transfer learning for this! So I am picking on a concept touched on by Tim Urban from one of his recent articles on waitbutwhy.com

Tim explains that before language was invented, every generation of humans had to re-invent the knowledge for themselves and this is how knowledge growth was happening from one generation to other:



Then, we invented language! A way to transfer learning from one generation to another and this is what happened over same time frame:

Tribal Knowledge Growth After Language

Beginner's guide to Web Scraping in Python (using BeautifulSoup) (<https://www.analyticsvidhya.com/blog/2015/10/beginner-guide-web-scraping-beautiful-soup-python/>)



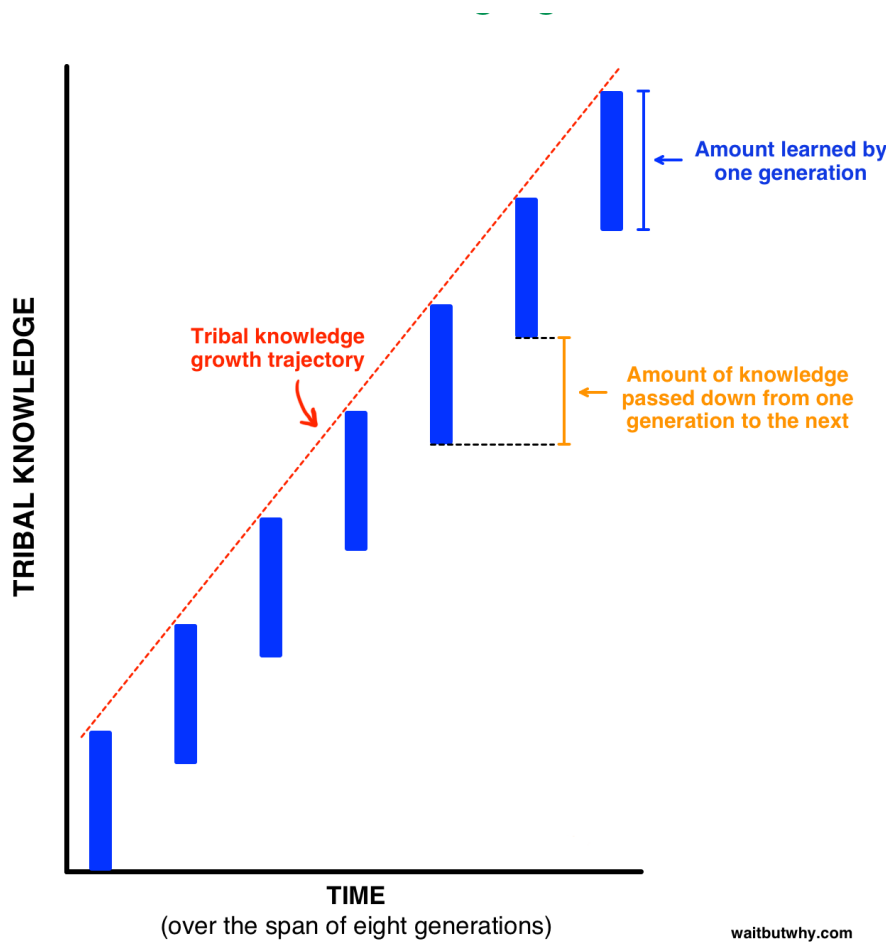
(https://www.dataquest.io/?utm_source=analytics-vidhya&utm_medium=ads&utm_campaign=2017-09-analytics-vidhya&utm_content=data-scientist)

RECENT POSTS



(<https://www.analyticsvidhya.com/blog/2017/10/comprehensive-tutorial-learn-data-science-julia-from-scratch/>)

**A
Comprehensive
Tutorial to Learn
Data Science
with Julia from**



Isn't it phenomenal and super empowering? So, transfer learning by passing on weights is equivalent of language used to disseminate knowledge over generations in human evolution.

What is a Pre-trained Model?

Simply put, a pre-trained model is a model created by some one else to solve a similar problem. Instead of building a model from scratch to solve a similar problem, you use the model trained on other problem as a starting point.

For example, if you want to build a self learning car. You can spend years to build a decent image recognition algorithm from scratch or you can take inception model (a pre-trained

Scratch
[\(https://www.analyticsvidhya.com/blog/2017/10/comprehensive-tutorial-learn-data-science-julia-from-scratch/\)](https://www.analyticsvidhya.com/blog/2017/10/comprehensive-tutorial-learn-data-science-julia-from-scratch/)
 MOHD SANAD ZAKI ...



[\(https://www.analyticsvidhya.com/blog/2017/10/essential-nlp-guide-data-scientists-top-10-nlp-tasks/\)](https://www.analyticsvidhya.com/blog/2017/10/essential-nlp-guide-data-scientists-top-10-nlp-tasks/)

The Essential NLP Guide for data scientists (with codes for top 10 common NLP tasks)
[\(https://www.analyticsvidhya.com/blog/2017/10/essential-nlp-guide-data-scientists-top-10-nlp-tasks/\)](https://www.analyticsvidhya.com/blog/2017/10/essential-nlp-guide-data-scientists-top-10-nlp-tasks/)

NSS , OCTOBER 26, ...



[\(https://www.analyticsvidhya.com/blog/2017/10/fundamentals-deep-learning-activation-](https://www.analyticsvidhya.com/blog/2017/10/fundamentals-deep-learning-activation-)

model) from Google which was built on ImageNet data to identify images in those pictures.

A pre-trained model may not be 100% accurate in your application, but it saves huge efforts required to re-invent the wheel. Let me show this to you with a recent example.

Why would we use Pre-trained Models?

I spent my last week working on a problem at CrowdAnalytix platform (<https://www.crowdanalytix.com/contests/identifying-themes-from-mobile-case-images>) – Identifying themes from mobile case images. This was an image classification problem where we were given 4591 images in the training dataset and 1200 images in the test dataset. The objective was to classify the images into one of the 16 categories. After the basic pre-processing steps, I started off with a simple MLP model (<https://www.analyticsvidhya.com/blog/2017/05/neural-network-from-scratch-in-python-and-r/>) with the following architecture-

Layer (type)	Output Shape	Param #
flatten_4 (Flatten)	(None, 150528)	0
dense_9 (Dense)	(None, 500)	75264500
activation_9 (Activation)	(None, 500)	0
dropout_7 (Dropout)	(None, 500)	0
dense_10 (Dense)	(None, 500)	250500
activation_10 (Activation)	(None, 500)	0
dropout_8 (Dropout)	(None, 500)	0
dense_11 (Dense)	(None, 500)	250500
activation_11 (Activation)	(None, 500)	0
dropout_9 (Dropout)	(None, 500)	0
dense_12 (Dense)	(None, 16)	8016
activation_12 (Activation)	(None, 16)	0

functions-when-to-use-them/)

Fundamentals of Deep Learning – Activation Functions and When to Use Them?

(<https://www.analyticsvidhya.com/blog/2017/10/fundamentals-deep-learning-activation-functions-when-to-use-them/>)

DISHASHREE GUPTA...



(<https://www.analyticsvidhya.com/blog/2017/10/art-story-telling-data-science/>)

The Art of Story Telling in Data Science and how to create data stories?

(<https://www.analyticsvidhya.com/blog/2017/10/art-story-telling-data-science/>)

SHANTANU KUMAR ,...

```
=====
Total params: 75,773,516
Trainable params: 75,773,516
Non-trainable params: 0
```

To simplify the above architecture after flattening the input image [224 X 224 X 3] into [150528], I used three hidden layers with 500, 500 and 500 neurons respectively. The output layer had 16 neurons which correspond to the number of categories in which we need to classify the input image.

I barely managed a training accuracy of 6.8 % which turned out to be very bad. Even experimenting with hidden layers, number of neurons in hidden layers and drop out rates. I could not manage to substantially increase my training accuracy. Increasing the hidden layers and the number of neurons, caused 20 seconds to run a single epoch on my Titan X GPU with 12 GB VRAM.

Below is an output of the training using the MLP model with the above architecture.

Epoch 10/10

50/50 [=====] - 21s - loss: 15.0100 - acc: 0.0688

As, you can see MLP was not going to give me any better results without exponentially increasing my training time. So I switched to Convolutional Neural Network to see how they perform on this dataset and whether I would be able to increase my training accuracy.

The CNN had the below architecture -

Layer (type)	Output Shape	Param #
conv2d_7 (Conv2D)	(None, 220, 220, 32)	2432
activation_27 (Activation)	(None, 220, 220, 32)	0
max_pooling2d_7 (MaxPooling2)	(None, 55, 55, 32)	0



(https://www.dataquest.io/?utm_source=analytics-vidhya&utm_medium=ads&utm_campaign=2017-09-analytics-vidhya&utm_content=python)



(http://www.edvancer.in/certified-data-scientist-with-python-course?utm_source=AV&utm_medium=AVads&utm_campaign=AVadsnonfc&utm_content=pythonavad)

GET CONNECTED



11,789

FOLLOWERS

(<http://www.twitter.com>

/analyticsvidhya)



39,815

FOLLOWERS

conv2d_8 (Conv2D)	(None, 51, 51, 32)	25632
activation_28 (Activation)	(None, 51, 51, 32)	0
max_pooling2d_8 (MaxPooling2	(None, 12, 12, 32)	0
conv2d_9 (Conv2D)	(None, 8, 8, 64)	51264
activation_29 (Activation)	(None, 8, 8, 64)	0
max_pooling2d_9 (MaxPooling2	(None, 2, 2, 64)	0
flatten_8 (Flatten)	(None, 256)	0
dense_21 (Dense)	(None, 64)	16448
activation_30 (Activation)	(None, 64)	0
dropout_12 (Dropout)	(None, 64)	0
dense_22 (Dense)	(None, 16)	1040
activation_31 (Activation)	(None, 16)	0
=====		
Total params: 96,816		
Trainable params: 96,816		
Non-trainable params: 0		

I used 3 convolutional blocks with each block following the below architecture-

1. 32 filters of size 5 X 5
2. Activation function – relu
3. Max pooling layer of size 4 X 4

The result obtained after the final convolutional block was flattened into a size [256] and passed into a single hidden layer of with 64 neurons. The output of the hidden layer was passed onto the output layer after a drop out rate of 0.5.

The result obtained with the above architecture is summarized below-

Epoch 10/10

50/50 [=====] – 21s – loss: 13.5733 – acc: 0.1575

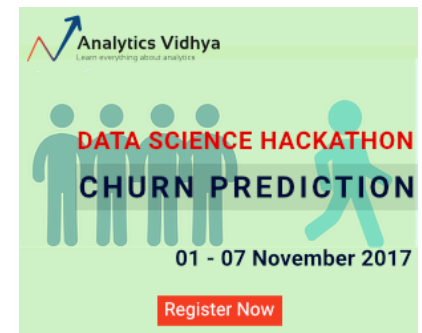
(<http://www.facebook.com/AnalyticsVidhya>)
2,248 FOLLOWERS

(<https://plus.google.com/+AnalyticsVidhya>)
Email SUBSCRIBE

(<http://feedburner.google.com/fb/a/mailverify?uri=analyticsvidhya>)



(https://www.analyticsvidhya.com/datahacksummit/?utm_source=avblog_side&utm_medium=web&utm_campaign=banner)



(https://datahack.analyticsvidhya.com/contest/data-science-hackathon-churn-prediction/?utm_source=avblog_side2&utm_medium=web&utm_campaign=banner)

Though my accuracy increased in comparison to the MLP output, it also increased the time taken to run a single epoch – 21 seconds.

But the major point to note was that the majority class in the dataset was around 17.6%. So, even if we had predicted the class of every image in the train dataset to be the majority class, we would have performed better than MLP and CNN respectively. Addition of more convolutional blocks substantially increased my training time. This led me to switch onto using pre-trained models where I would not have to train my entire architecture but only a few layers.

So, I used VGG16 model which is pre-trained on the ImageNet dataset and provided in the keras library for use. Below is the architecture of the VGG16 model which I used.

Layer (type)	Output Shape	Param #
input_1 (InputLayer)	(None, 224, 224, 3)	0
block1_conv1 (Conv2D)	(None, 224, 224, 64)	1792
block1_conv2 (Conv2D)	(None, 224, 224, 64)	36928
block1_pool (MaxPooling2D)	(None, 112, 112, 64)	0
block2_conv1 (Conv2D)	(None, 112, 112, 128)	73856
block2_conv2 (Conv2D)	(None, 112, 112, 128)	147584
block2_pool (MaxPooling2D)	(None, 56, 56, 128)	0
block3_conv1 (Conv2D)	(None, 56, 56, 256)	295168
block3_conv2 (Conv2D)	(None, 56, 56, 256)	590080
block3_conv3 (Conv2D)	(None, 56, 56, 256)	590080
block3_pool (MaxPooling2D)	(None, 28, 28, 256)	0
block4_conv1 (Conv2D)	(None, 28, 28, 512)	1180160
block4_conv2 (Conv2D)	(None, 28, 28, 512)	2359808
block4_conv3 (Conv2D)	(None, 28, 28, 512)	2359808
block4_pool (MaxPooling2D)	(None, 14, 14, 512)	0
block5_conv1 (Conv2D)	(None, 14, 14, 512)	2359808

block5_conv2 (Conv2D)	(None, 14, 14, 512)	2359808
block5_conv3 (Conv2D)	(None, 14, 14, 512)	2359808
block5_pool (MaxPooling2D)	(None, 7, 7, 512)	0
flatten (Flatten)	(None, 25088)	0
fc1 (Dense)	(None, 4096)	102764544
fc2 (Dense)	(None, 4096)	16781312
predictions (Dense)	(None, 1000)	4097000
dense_1 (Dense)	(None, 16)	16016
=====		
Total params: 138,373,560		
Trainable params: 138,373,560		
Non-trainable params: 0		

The only change that I made to the VGG16 existing architecture is changing the softmax layer with 1000 outputs to 16 categories suitable for our problem and re-training the dense layer.

This architecture gave me an accuracy of 70% much better than MLP and CNN. Also, the biggest benefit of using the VGG16 pre-trained model was almost negligible time to train the dense layer with greater accuracy.

So, I moved forward with this approach of using a pre-trained model and the next step was to fine tune my VGG16 model to suit this problem.

How can I use Pre-trained Models?

What is our objective when we train a neural network? We wish to identify the correct weights for the network by multiple forward and backward iterations. By using pre-trained models which have been previously trained on large datasets, we can directly use the weights and architecture obtained and apply the learning on our problem statement. This is known as transfer learning. We "transfer the learning" of the pre-trained

model to our specific problem statement.

You should be very careful while choosing what pre-trained model you should use in your case. If the problem statement we have at hand is very different from the one on which the pre-trained model was trained – the prediction we would get would be very inaccurate. For example, a model previously trained for speech recognition would work horribly if we try to use it to identify objects using it.

We are lucky that many pre-trained architectures are directly available for us in the Keras library. **Imagenet** data set has been widely used to build various architectures since it is large enough (1.2M images) to create a generalized model. The problem statement is to train a model that can correctly classify the images into 1,000 separate object categories. These 1,000 image categories represent object classes that we come across in our day-to-day lives, such as species of dogs, cats, various household objects, vehicle types etc.

These pre-trained networks demonstrate a strong ability to generalize to images outside the ImageNet dataset via transfer learning. We make modifications in the pre-existing model by fine-tuning the model. Since we assume that the pre-trained network has been trained quite well, we would not want to modify the weights too soon and too much. While modifying we generally use a learning rate smaller than the one used for initially training the model.

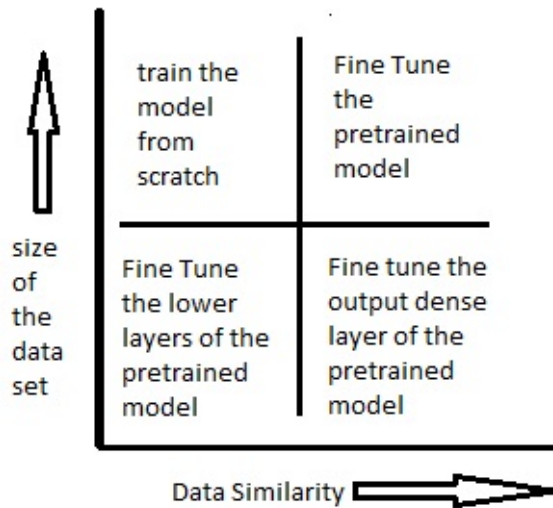
Ways to Fine tune the model

1. **Feature extraction** – We can use a pre-trained model as a feature extraction mechanism. What we can do is that we can remove the output layer(the one which gives the probabilities for being in each of the 1000 classes) and then use the entire

network as a fixed feature extractor for the new data set.

2. **Use the Architecture of the pre-trained model** – What we can do is that we use architecture of the model while we initialize all the weights randomly and train the model according to our dataset again.
3. **Train some layers while freeze others** – Another way to use a pre-trained model is to train is partially. What we can do is we keep the weights of initial layers of the model frozen while we retrain only the higher layers. We can try and test as to how many layers to be frozen and how many to be trained.

The below diagram should help you decide on how to proceed on using the pre trained model in your case –



Scenario 1 – Size of the Data set is small while the Data

similarity is very high – In this case, since the data similarity is very high, we do not need to retrain the model. All we need to do is to customize and modify the output layers according to our problem statement. We use the pretrained model as a feature extractor. Suppose we decide to use models trained on Imagenet to identify if the new set of images have cats or dogs. Here the images we need to identify would be similar to imagenet, however we just need two categories as my output – cats or dogs. In this case all we do is just modify the dense

layers and the final softmax layer to output 2 categories instead of a 1000.

Scenario 2 – Size of the data is small as well as data similarity

is very low – In this case we can freeze the initial (let's say k) layers of the pretrained model and train just the remaining $(n-k)$ layers again. The top layers would then be customized to the new data set. Since the new data set has low similarity it is significant to retrain and customize the higher layers according to the new dataset. The small size of the data set is compensated by the fact that the initial layers are kept pretrained (which have been trained on a large dataset previously) and the weights for those layers are frozen.

Scenario 3 – Size of the data set is large however the Data

similarity is very low – In this case, since we have a large dataset, our neural network training would be effective. However, since the data we have is very different as compared to the data used for training our pretrained models. The predictions made using pretrained models would not be effective. Hence, it's best to train the neural network from scratch according to your data.

Scenario 4 – Size of the data is large as well as there is high

data similarity – This is the ideal situation. In this case the pretrained model should be most effective. The best way to use the model is to retain the architecture of the model and the initial weights of the model. Then we can retrain this model using the weights as initialized in the pre-trained model.

Use the pre-trained models to identify handwritten digits

Let's now try to use a pretrained model for a simple problem.

There are various architectures that have been trained on the imageNet data set. You can go through various architectures here (<http://www.pyimagesearch.com/2017/03/20/imagenet-vggnet-resnet-inception-xception-keras/>). I have used vgg16 as pretrained model architecture and have tried to identify handwritten digits using it. Let's see in which of the above scenarios would this problem fall into. We have around 60,000 training images of handwritten digits. This data set is definitely small. So the situation would either fall into scenario 1 or scenario 2. We shall try to solve the problem using both these scenarios. The data set can be downloaded from here (<http://yann.lecun.com/exdb/mnist/>).

1. **Retrain the output dense layers only** – Here we use vgg16 as a feature extractor. We then use these features and send them to dense layers which are trained according to our data set. The output layer is also replaced with our new softmax layer relevant to our problem. The output layer in a vgg16 is a softmax activation with 1000 categories. We remove this layer and replace it with a softmax layer of 10 categories. We just train the weights of these layers and try to identify the digits.

```
# importing required libraries

from keras.models import Sequential
from scipy.misc import imread
get_ipython().magic('matplotlib inline')
import matplotlib.pyplot as plt
import numpy as np
import keras
from keras.layers import Dense
import pandas as pd

from keras.applications.vgg16 import VGG16
from keras.preprocessing import image
```

```
from keras.applications.vgg16 import preprocess_input
import numpy as np
from keras.applications.vgg16 import
decode_predictions
train=pd.read_csv("R/Data/Train/train.csv")
test=pd.read_csv("R/Data/test.csv")
train_path="R/Data/Train/Images/train/"
test_path="R/Data/Train/Images/test/"

from scipy.misc import imread
# preparing the train dataset

train_img=[]
for i in range(len(train)):

    temp_img=image.load_img(train_path+train['filename']
[i],target_size=(224,224))

    temp_img=image.img_to_array(temp_img)

    train_img.append(temp_img)

#converting train images to array and applying mean
subtraction processing

train_img=np.array(train_img)
train_img=preprocess_input(train_img)
# applying the same procedure with the test dataset

test_img=[]
for i in range(len(test)):

    temp_img=image.load_img(test_path+test['filename']
[i],target_size=(224,224))
```

```
temp_img=image.img_to_array(temp_img)

test_img.append(temp_img)

test_img=np.array(test_img)
test_img=preprocess_input(test_img)

# loading VGG16 model weights
model = VGG16(weights='imagenet', include_top=False)
# Extracting features from the train dataset using
the VGG16 pre-trained model

features_train=model.predict(train_img)
# Extracting features from the train dataset using
the VGG16 pre-trained model

features_test=model.predict(test_img)

# flattening the layers to conform to MLP input

train_x=features_train.reshape(49000,25088)
# converting target variable to array

train_y=np.asarray(train['label'])
# performing one-hot encoding for the target variable

train_y=pd.get_dummies(train_y)
train_y=np.array(train_y)
# creating training and validation set

from sklearn.model_selection import train_test_split
X_train, X_valid, Y_train,
Y_valid=train_test_split(train_x,train_y,test_size=0.3,
random_state=42)
```

```
# creating a mlp model
from keras.layers import Dense, Activation
model=Sequential()

model.add(Dense(1000, input_dim=25088,
activation='relu',kernel_initializer='uniform'))
keras.layers.core.Dropout(0.3, noise_shape=None,
seed=None)

model.add(Dense(500,input_dim=1000,activation='sigmoid'))
keras.layers.core.Dropout(0.4, noise_shape=None,
seed=None)

model.add(Dense(150,input_dim=500,activation='sigmoid'))
keras.layers.core.Dropout(0.2, noise_shape=None,
seed=None)

model.add(Dense(units=10))
model.add(Activation('softmax'))

model.compile(loss='categorical_crossentropy',
optimizer="adam", metrics=['accuracy'])

# fitting the model

model.fit(X_train, Y_train, epochs=20,
batch_size=128,validation_data=(X_valid,Y_valid))
```

2. Freeze the weights of first few layers – Here what we do is we freeze the weights of the first 8 layers of the vgg16 network, while we retrain the subsequent layers. This is because the first

few layers capture universal features like curves and edges that are also relevant to our new problem. We want to keep those weights intact and we will get the network to focus on learning dataset-specific features in the subsequent layers.

Code for freezing the weights of first few layers.

```
from keras.models import Sequential
from scipy.misc import imread
get_ipython().magic('matplotlib inline')
import matplotlib.pyplot as plt
import numpy as np
import keras
from keras.layers import Dense
import pandas as pd

from keras.applications.vgg16 import VGG16
from keras.preprocessing import image
from keras.applications.vgg16 import preprocess_input
import numpy as np
from keras.applications.vgg16 import
decode_predictions
from keras.utils.np_utils import to_categorical

from sklearn.preprocessing import LabelEncoder
from keras.models import Sequential
from keras.optimizers import SGD
from keras.layers import Input, Dense, Convolution2D,
MaxPooling2D, AveragePooling2D, ZeroPadding2D,
Dropout, Flatten, merge, Reshape, Activation

from sklearn.metrics import log_loss
```

```
train=pd.read_csv("R/Data/Train/train.csv")
test=pd.read_csv("R/Data/test.csv")
train_path="R/Data/Train/Images/train/"
test_path="R/Data/Train/Images/test/"

from scipy.misc import imread

train_img=[]
for i in range(len(train)):

    temp_img=imread(train_path+train['filename']
[i],target_size=(224,224))

    temp_img=imread(temp_img)

    train_img.append(temp_img)

train_img=np.array(train_img)
train_img=preprocess_input(train_img)

test_img=[]
for i in range(len(test)):

    temp_img=imread(test_path+test['filename']
[i],target_size=(224,224))

    temp_img=imread(temp_img)

    test_img.append(temp_img)

test_img=np.array(test_img)
test_img=preprocess_input(test_img)
```



```
from keras.models import Model

def vgg16_model(img_rows, img_cols, channel=1,
num_classes=None):

    model = VGG16(weights='imagenet',
include_top=True)

    model.layers.pop()

    model.outputs = [model.layers[-1].output]

    model.layers[-1].outbound_nodes = []

    x=Dense(num_classes, activation='softmax')
(model.output)

    model=Model(model.input,x)

    #To set the first 8 layers to non-trainable (weights
will not be updated)

    for layer in model.layers[:8]:

        layer.trainable = False

    # Learning rate is changed to 0.001
    sgd = SGD(lr=1e-3, decay=1e-6, momentum=0.9,
nesterov=True)
    model.compile(optimizer=sgd,
loss='categorical_crossentropy', metrics=
['accuracy'])

    return model

train_y=np.asarray(train['label'])
```

```
le = LabelEncoder()

train_y = le.fit_transform(train_y)

train_y=to_categorical(train_y)

train_y=np.array(train_y)

from sklearn.model_selection import train_test_split
X_train, X_valid, Y_train,
Y_valid=train_test_split(train_img,train_y,test_size=0.2,
random_state=42)

# Example to fine-tune on 3000 samples from Cifar10

img_rows, img_cols = 224, 224 # Resolution of inputs
channel = 3
num_classes = 10
batch_size = 16
nb_epoch = 10

# Load our model
model = vgg16_model(img_rows, img_cols, channel,
num_classes)

model.summary()
# Start Fine-tuning
model.fit(X_train,
Y_train,batch_size=batch_size,epochs=nb_epoch,shuffle=True,verbose=1,validation_data=
(X_valid, Y_valid))


# Make predictions
predictions_valid = model.predict(X_valid,
batch_size=batch_size, verbose=1)
```

```
# Cross-entropy loss score  
score = log_loss(Y_valid, predictions_valid)
```


End Notes


I hope that you would now be able to apply pre-trained models to your problem statements. Be sure that the pre-trained model you have selected has been trained on a similar data set as the one that you wish to use it on. There are various architectures people have tried on different types of data sets and I strongly encourage you to go through these architectures and apply them on your own problem statements. Please feel free to discuss your doubts and concerns in the comments section.

Share this:


 (<https://www.analyticsvidhya.com/blog/2017/06/transfer-learning-the-art-of-fine-tuning-a-pre-trained-model/?share=linkedin&nb=1>) 186

 (<https://www.analyticsvidhya.com/blog/2017/06/transfer-learning-the-art-of-fine-tuning-a-pre-trained-model/?share=facebook&nb=1>) 244

 (<https://www.analyticsvidhya.com/blog/2017/06/transfer-learning-the-art-of-fine-tuning-a-pre-trained-model/?share=google-plus-1&nb=1>)

 (<https://www.analyticsvidhya.com/blog/2017/06/transfer-learning-the-art-of-fine-tuning-a-pre-trained-model/?share=twitter&nb=1>)

 (<https://www.analyticsvidhya.com/blog/2017/06/transfer-learning-the-art-of-fine-tuning-a-pre-trained-model/?share=pocket&nb=1>)

 (<https://www.analyticsvidhya.com/blog/2017/06/transfer-learning-the-art-of-fine-tuning-a-pre-trained-model/?share=reddit&nb=1>)



(<https://www.analyticsvidhya.com/blog/2017/05/gpus-necessary-for-deep-learning/>)

Why are GPUs necessary for training Deep Learning models?

(<https://www.analyticsvidhya.com/blog/2017/05/gpus-necessary-for-deep-learning/>)

May 18, 2017

In "Deep Learning"



(<https://www.analyticsvidhya.com/blog/2015/06/machine-learning-basics/>)

Machine Learning basics for a newbie

(<https://www.analyticsvidhya.com/blog/2015/06/machine-learning-basics/>)

June 11, 2015

In "Business Analytics"

In "Machine Learning"

(<https://www.analyticsvidhya.com/blog/2017/01/must-know-questions-deep-learning/>)

45 Questions to test a data scientist on basics of Deep Learning (along with solution)

(<https://www.analyticsvidhya.com/blog/2017/01/must-know-questions-deep-learning/>)

January 29, 2017

In "Machine Learning"

RELATED

TAGS: DEEP LEARNING

(<https://www.analyticsvidhya.com/blog/tag/deep-learning/>), DEEP

LEARNING IN PYTHON

(<https://www.analyticsvidhya.com/blog/tag/deep-learning-in-python/>), INCEPTION

(<https://www.analyticsvidhya.com/blog/tag/inception/>), KERAS

(<https://www.analyticsvidhya.com/blog/tag/keras/>), PARAMETER TUNING

(<https://www.analyticsvidhya.com/blog/tag/parameter-tuning/>), PRE-TRAINED MODELS

(<https://www.analyticsvidhya.com/blog/tag/pre-trained-models/>), VGG16

(<https://www.analyticsvidhya.com/blog/tag/vgg16/>)

Previous Article



Data Scientist -Mumbai
(1-4 Years Of
Experience)

(<https://www.analyticsvidhya.com/blog/2017/06/data-scientist-mumbai-1-4-years-of-experience/>)

Next Article



Data Science Evangelist-
Gurgaon (2 to 3 years of
experience)

(<https://www.analyticsvidhya.com/blog/2017/06/data-science-evangelist-gurgaon-2-to-3-years-of-experience/>)



(<https://www.analyticsvidhya.com/blog/author/dishashree26/>)

Author

Dishashree Gupta
(<https://www.analyticsvidhya.com/blog/author/dishashree26/>)

Dishashree is passionate about statistics and is a machine learning enthusiast. She has an experience of 1.5 years of Market Research using R, advanced Excel, Azure ML.

17 COMMENTS

Ashish Mokalkar says:
REPLY (HTTPS://WWW.ANALYTICSVIDHYA.COM/BLOG/2017/06/TRANSFER-LEARNING-THE-ART-OF-FINE-TUNING-A-PRE-TRAINED-MODEL/#COMMENT-129584#RESPONSE) (HTTPS://WWW.ANALYTICSVIDHYA.COM/BLOG/2017/06/TRANSFER-LEARNING-THE-ART-OF-FINE-TUNING-A-PRE-TRAINED-MODEL/#COMMENT-129584)

Hey! You havent given folder structure and format in which training data is stored. People wishing to try out this example may not be having idea where to download training data from and where to place it. It would elucidate them if you update the same.



Dishashree Gupta says:
REPLY (HTTPS://WWW.ANALYTICSVIDHYA.COM/BLOG/2017/06/TRANSFER-LEARNING-THE-ART-OF-FINE-TUNING-A-PRE-TRAINED-MODEL/#COMMENT-129686) JUNE 8, 2017 AT 09:11 PM (RESPOND)
/BLOG/2017/06/TRANSFER-LEARNING-THE-ART-OF-FINE-TUNING-A-PRE-TRAINED-MODEL/#COMMENT-129686)

I have added the link for the download of MNIST data.



Shamane says:
REPLY (HTTPS://WWW.ANALYTICSVIDHYA.COM/BLOG/2017/06/TRANSFER-LEARNING-THE-ART-OF-FINE-TUNING-A-PRE-TRAINED-MODEL/#COMMENT-129611) JUNE 8, 2017 AT 06:11 PM (RESPOND)
/BLOG/2017/06/TRANSFER-LEARNING-THE-ART-OF-FINE-TUNING-A-PRE-TRAINED-MODEL/#COMMENT-129611)

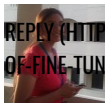
perfect



Gouthap says:
REPLY (HTTPS://WWW.ANALYTICSVIDHYA.COM/BLOG/2017/06/TRANSFER-LEARNING-THE-ART-OF-FINE-TUNING-A-PRE-TRAINED-MODEL/#COMMENT-129675) JUNE 8, 2017 AT 06:15 PM (RESPOND)
/BLOG/2017/06/TRANSFER-LEARNING-THE-ART-OF-FINE-TUNING-A-PRE-TRAINED-MODEL/#COMMENT-129675)

Same here, what does the CSV files contain? please explain why the train and test data was loaded twice esp. why it is required to load while freezing the weights?

appreciate your response..



Dishashree Gupta says:
REPLY (HTTPS://WWW.ANALYTICSVIDHYA.COM/BLOG/2017/06/TRANSFER-LEARNING-THE-ART-OF-FINE-TUNING-A-PRE-TRAINED-MODEL/#COMMENT-129687) JUNE 8, 2017 AT 09:11 PM (RESPOND)
/BLOG/2017/06/TRANSFER-LEARNING-THE-ART-OF-FINE-TUNING-A-PRE-TRAINED-MODEL/#COMMENT-129687)

So I have added the link of MNIST data. The csv file for train consists the name of images and the corresponding digits for them.

The test and train files have been loaded just because I wanted to keep the two codes exclusive.

Analyticsvidhya248@laposte.net says:
REPLY (HTTPS://WWW.ANALYTICSVIDHYA.COM/BLOG/2017/06/TRANSFER-LEARNING-THE-ART-OF-FINE-TUNING-A-PRE-TRAINED-MODEL/?REPLYTOCOM=131676#RESPOND) //WWW.ANALYTICSVIDHYA.COM/BLOG/2017/06/TRANSFER-LEARNING-THE-ART-OF-FINE-TUNING-A-PRE-TRAINED-MODEL/#COMMENT-131676)

Hi,

Your presentation is very nice. But my case is that I just begin to deal with image classification with CNN. I Know how to convert mnist data to csv file since it's a 1-D image.

I really need help to know how to convert a 3-Dimensional images into csv file.

Thanks,

Kai says:
REPLY (HTTPS://WWW.ANALYTICSVIDHYA.COM/BLOG/2017/06/TRANSFER-LEARNING-THE-ART-OF-FINE-TUNING-A-PRE-TRAINED-MODEL/?REPLYTOCOM=129884#RESPOND) //WWW.ANALYTICSVIDHYA.COM/BLOG/2017/06/TRANSFER-LEARNING-THE-ART-OF-FINE-TUNING-A-PRE-TRAINED-MODEL/#COMMENT-129884)

Thanks for your sharing.

Quick question:

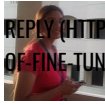
1. Can we substract the data from the mean of our data for each channel instead of from the mean of vgg16 data? What is the difference for this two preprocess?
2. What if I want to train by grayscale image?

Kai says:
REPLY (HTTPS://WWW.ANALYTICSVIDHYA.COM/BLOG/2017/06/TRANSFER-LEARNING-THE-ART-OF-FINE-TUNING-A-PRE-TRAINED-MODEL/?REPLYTOCOM=129994#RESPOND) //WWW.ANALYTICSVIDHYA.COM/BLOG/2017/06/TRANSFER-LEARNING-THE-ART-OF-FINE-TUNING-A-PRE-TRAINED-MODEL/#COMMENT-129994)

Hi!

Should we apply the same preprocess function from the network we transfer on our data?
i.e. substract training data from the mean of our data or

the mean of transfer network's data?



Dishashree Gupta says
REPLY (HTTPS://WWW.ANALYTICSVIDHYA.COM/BLOG/2017/06/TRANSFER-LEARNING-THE-ART-OF-FINE-TUNING-A-PRE-TRAINED-MODEL/#COMMENT-130150#RESPOND)
JUNE 8, 2017 AT 6:58 PM (REPLY TO COM-130150#RESPOND)
/BLOG/2017/06/TRANSFER-LEARNING-THE-ART-OF-FINE-TUNING-A-PRE-TRAINED-MODEL/#COMMENT-130150)

Hi Kai,

So the preprocess function is subtracting the mean of your own dataset rather than the data of the pretrained.

It doesn't mean much to actually subtract the mean of the network's data since it might be very different than your own.



Rico says
REPLY (HTTPS://WWW.ANALYTICSVIDHYA.COM/BLOG/2017/06/TRANSFER-LEARNING-THE-ART-OF-FINE-TUNING-A-PRE-TRAINED-MODEL/#COMMENT-130029#RESPOND)
JUNE 8, 2017 AT 4:28 PM (REPLY TO COM-130029#RESPOND)
/BLOG/2017/06/TRANSFER-LEARNING-THE-ART-OF-FINE-TUNING-A-PRE-TRAINED-MODEL/#COMMENT-130029)

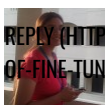
Dear Dishashree.

When trying to apply this to my own dataset, the program throws the error ValueError: cannot reshape array of size 5143040 into shape (49000,25088).

I guess this is because of the size difference in the datasets.

Can you explain what the numbers 49000 and 25088 mean?

Best regards



Dishashree Gupta says
REPLY (HTTPS://WWW.ANALYTICSVIDHYA.COM/BLOG/2017/06/TRANSFER-LEARNING-THE-ART-OF-FINE-TUNING-A-PRE-TRAINED-MODEL/#COMMENT-130149#RESPOND)
JUNE 8, 2017 AT 6:58 PM (REPLY TO COM-130149#RESPOND)
/BLOG/2017/06/TRANSFER-LEARNING-THE-ART-OF-FINE-TUNING-A-PRE-TRAINED-MODEL/#COMMENT-130149)

Hi Rico,

the numbers 49000 and 25088 are the dimensions of your data set. So you would have 49000 records with each record having 25088 features. You should check your reshaping function. Since the error lies there !

Sudipto Banerjee says:
REPLY (HTTPS://WWW.ANALYTICSVIDHYA.COM/BLOG/2017/06/TRANSFER-LEARNING-THE-ART-OF-FINE-TUNING-A-PRE-TRAINED-MODEL/#COMMENT-130462#RESPONSE) (HTTPS://WWW.ANALYTICSVIDHYA.COM/BLOG/2017/06/TRANSFER-LEARNING-THE-ART-OF-FINE-TUNING-A-PRE-TRAINED-MODEL/#COMMENT-130462)

Hi, I would like to know what should I enter in ("R/Data /Train/train.csv"), ['filename'] and ['label']?
And my image size is 850*600. Where should I mention the size and what other specifications should I consider to accord with my image size?

```
train=pd.read_csv("R/Data/Train/train.csv")
test=pd.read_csv("R/Data/test.csv")
train_path="R/Data/Train/Images/train/"
test_path="R/Data/Train/Images/test/"
```

```
temp_img=image.load_img(train_path+train['filename']
[i],target_size=(224,224))
```

```
train_y=np.asarray(train['label'])
```

Robert M says:
REPLY (HTTPS://WWW.ANALYTICSVIDHYA.COM/BLOG/2017/06/TRANSFER-LEARNING-THE-ART-OF-FINE-TUNING-A-PRE-TRAINED-MODEL/#COMMENT-130485#RESPONSE) (HTTPS://WWW.ANALYTICSVIDHYA.COM/BLOG/2017/06/TRANSFER-LEARNING-THE-ART-OF-FINE-TUNING-A-PRE-TRAINED-MODEL/#COMMENT-130485)

Hello,
Thank you for posting this. I have though some problems in implementing your algorithm:

1) what is train['filename'][i] ? where is defined in the code?
I just downloaded the csv files from your ink. Did I missed something?

2) why target_size=(224,224) when the images are 28×28?
Could you help me with this?
Thank you!

harish says
REPLY (HTTPS://WWW.ANALYTICSVIDHYA.COM/BLOG/2017/06/TRANSFER-LEARNING-THE-ART-OF-FINE-TUNING-A-PRE-TRAINED-MODEL/#COMMENT-130625#RESPOND) /WWW.ANALYTICSVIDHYA.COM/BLOG/2017/06/TRANSFER-LEARNING-THE-ART-OF-FINE-TUNING-A-PRE-TRAINED-MODEL/#COMMENT-130625)

awesome explanation

Akram Khan (http://www.codeverb.com) says
REPLY (HTTPS://WWW.ANALYTICSVIDHYA.COM/BLOG/2017/06/TRANSFER-LEARNING-THE-ART-OF-FINE-TUNING-A-PRE-TRAINED-MODEL/#COMMENT-130845#RESPOND) /WWW.ANALYTICSVIDHYA.COM/BLOG/2017/06/TRANSFER-LEARNING-THE-ART-OF-FINE-TUNING-A-PRE-TRAINED-MODEL/#COMMENT-130845)

This is a very informative article. I also agree with your post title and your really well explain your point of view. I am very happy to see this post. Thanks for share with us. Keep it up and share the more most related post. quiz online Programming test (<http://www.codeverb.com>)

Abram says
REPLY (HTTPS://WWW.ANALYTICSVIDHYA.COM/BLOG/2017/06/TRANSFER-LEARNING-THE-ART-OF-FINE-TUNING-A-PRE-TRAINED-MODEL/#COMMENT-131197#RESPOND) /WWW.ANALYTICSVIDHYA.COM/BLOG/2017/06/TRANSFER-LEARNING-THE-ART-OF-FINE-TUNING-A-PRE-TRAINED-MODEL/#COMMENT-131197)

Ma'am can you help me to solve the Identifying themes from mobile case images problem.
This problem has been assigned to me as project.
my e-mail-modig607@gmail.com (mailto:e-mail-modig607@gmail.com)

Fadhwa says
REPLY (HTTPS://WWW.ANALYTICSVIDHYA.COM/BLOG/2017/06/TRANSFER-LEARNING-THE-ART-OF-FINE-TUNING-A-PRE-TRAINED-MODEL/#COMMENT-132426#RESPOND) /WWW.ANALYTICSVIDHYA.COM/BLOG/2017/06/TRANSFER-LEARNING-THE-ART-OF-FINE-TUNING-A-PRE-TRAINED-MODEL/#COMMENT-132426)

Thank you for this amazing blog,

In the link of the MNSIT data. I did not find train.csv.
Could you provide us with some explanation for this?

LEAVE A REPLY

Your email address will not be published.

Comment

Name (required)

Email (required)

Website

SUBMIT COM



ANALYTICS VIDHYA

About Us
(http://www.analyticsvidhya.com/about-me/)

Our Team
(https://www.analyticsvidhya.com/about-me/team/)

Career
(https://www.analyticsvidhya.com/career-analytics-vidhya/)

Contact Us
(https://www.analyticsvidhya.com/contact/)

Write for us
(https://www.analyticsvidhya.com/about-me/write/)

DATA SCIENTISTS COMPANIES

Blog
(https://www.analyticsvidhya.com/blog/)

Hackathon
(https://datahack.analyticsvidhya.com/)

Discussions
(https://discuss.analyticsvidhya.com/)

Apply Jobs
(https://www.analyticsvidhya.com/jobs/)

Leaderboard
(https://datahack.analyticsvidhya.com/users/)

Post Jobs
(https://www.analyticsvidhya.com/corporate/)

Trainings
(https://datahack.analyticsvidhya.com/trainings/)

Analytics Hackathons
(https://datahack.analyticsvidhya.com/hackathons/)

Recruitment
(https://www.analyticsvidhya.com/contact/)

JOIN OUR COMMUNITY :

f

(https://www.facebook.com/AnalyticsVidhya/)

10821

(https://www.facebook.com/AnalyticsVidhya/)

tw

(https://twitter.com/AnalyticsVidhya)

10821

(https://twitter.com/AnalyticsVidhya)

g+

(https://plus.google.com/+Analyticsvidhya/)

2110

(https://plus.google.com/+Analyticsvidhya/)

in

(https://www.linkedin.com/company/Analyticsvidhya/)

2521

(https://www.linkedin.com/company/Analyticsvidhya/)

Followers

(https://plus.google.com/+Analyticsvidhya/)

Followers

(https://plus.google.com/+Analyticsvidhya/)

Subscribe to emailer

>