

Deep Learning : Image recognition

Capstone project #3

Goal : Build a Neural Network that recognizes objects in images

- Create a custom convolutional neural network model
- Used pre trained model VGG16
- Used transfer learning

Softwares : Python3, Jupyter Notebook

Libraries : Matplotlib, Numpy, Tensorflow

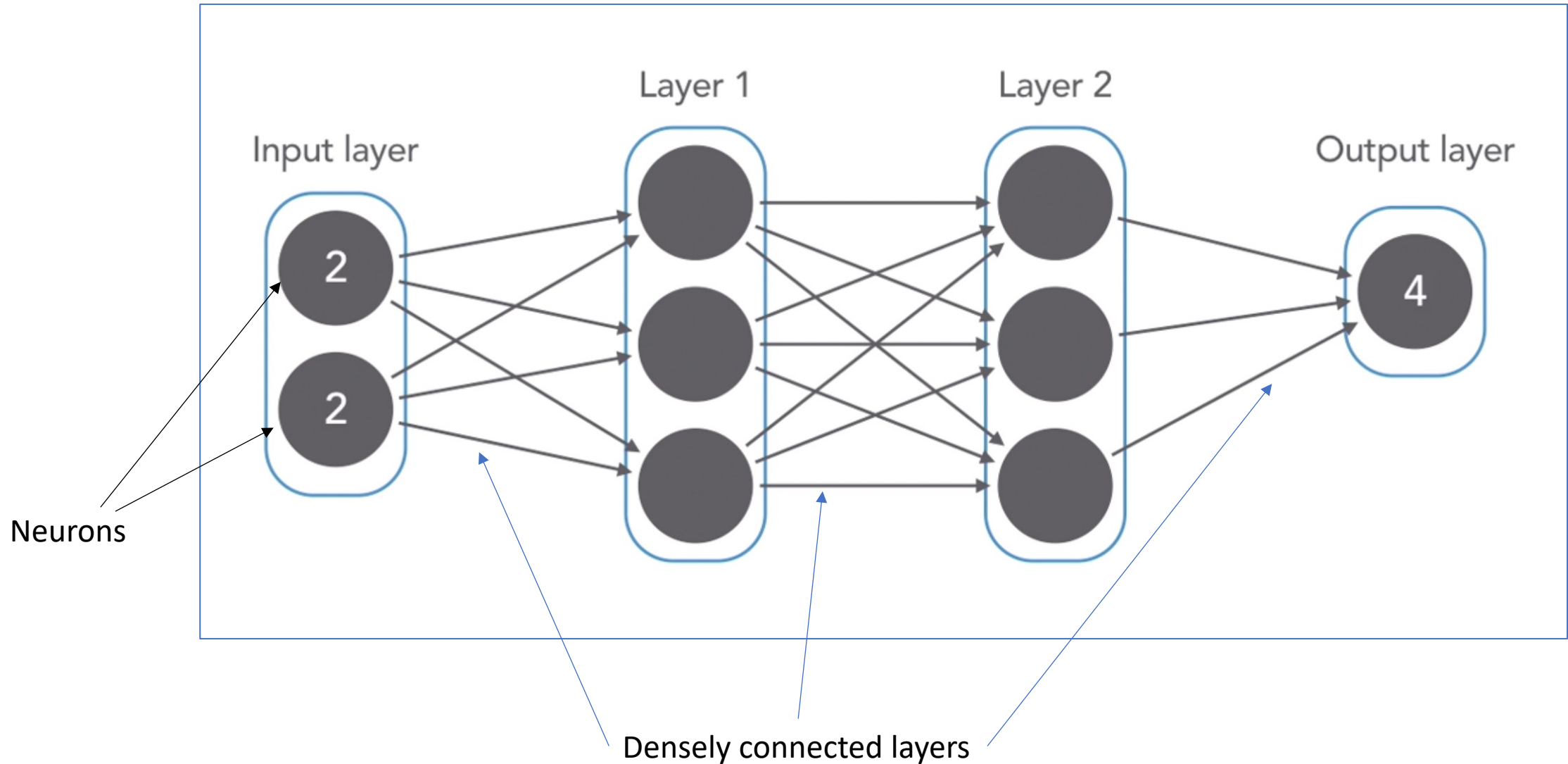
Dataset – Cifar-10, link : <https://www.cs.toronto.edu/~kriz/cifar.html>

This dataset consists of thousands of images of 10 different objects. Each image is labelled, so we know which kind of image it is. Using this dataset, we can train our NN to identify these objects

Structure

- Neural Network
- Image Classification
 - Architecture of Deep Neural Network
- Data exploration
- Create a Neural Network from scratch
- Use pre-trained model – VGG16
- Use Transfer Learning
- Comparison of neural networks created

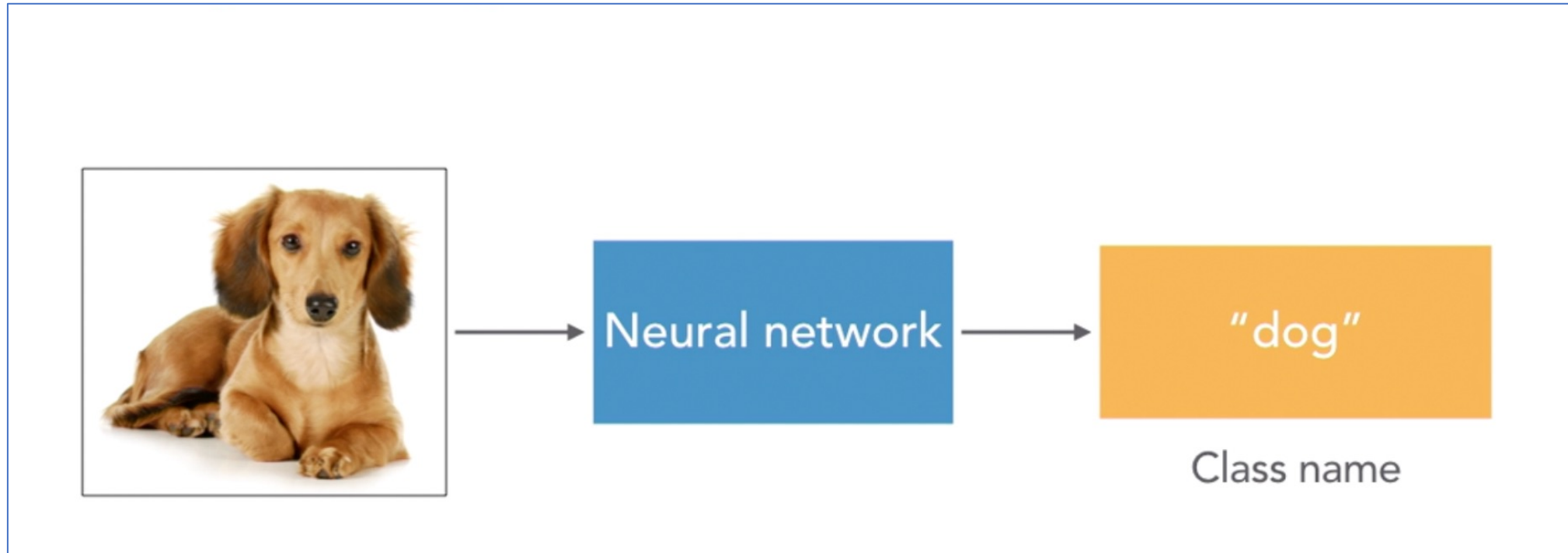
Simple Neural Network that takes two inputs and produces their sum as output



How Image Classification works ?

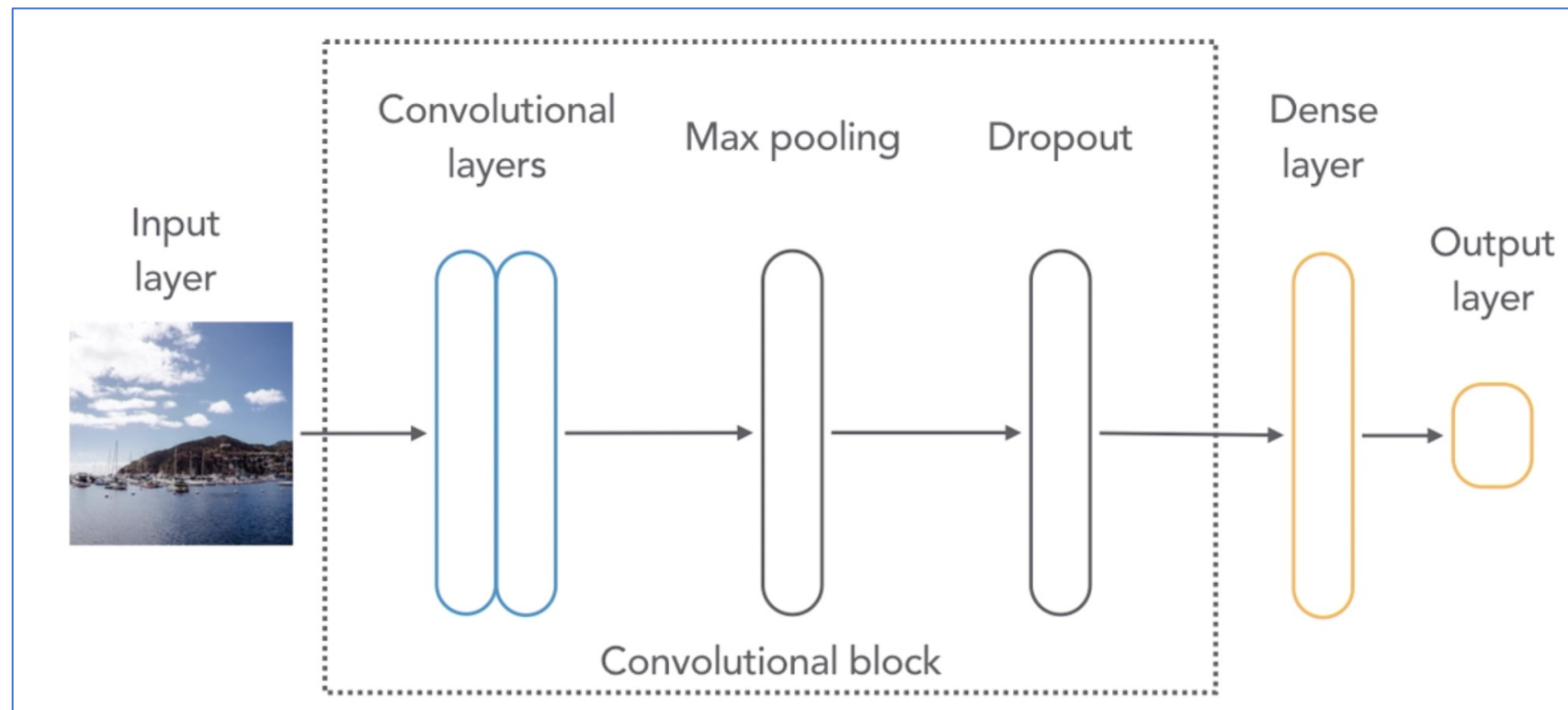
Definition : Ability for computers to look at a photograph and understand what is inside

Feed an image to a Neural Network which produces a dog label, as that is the main object that appears in the image

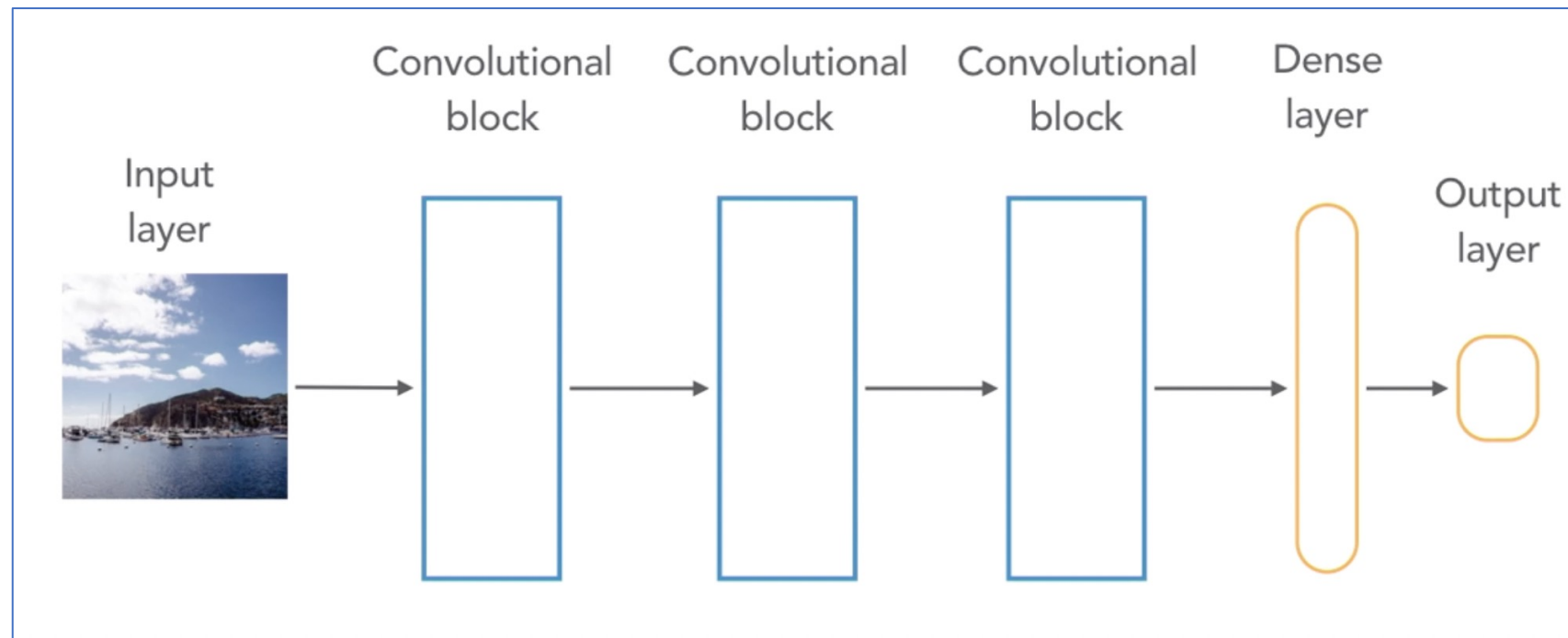


Deep Neural Network

Definition : Adding more layers to a Neural Network gives it more capacity to identify more complex patterns and shapes



- Convolutional layer is added to make it possible to detect a pattern in an image no matter where it is located in the image – Translational invariance
- The idea of Max Pooling is to down sample the data by passing on the most important bits only. The output will be the same though it will be less work for the Neural Network
- Dropout layer throws away some data randomly, forcing the Neural Network to try hard to learn and not memorize training data
- Dense layer maps the output of the previous layers to the output layer so that Neural Network can predict which class the input image belongs to



To make our Neural Network more powerful and be able to recognize more complex images, we can add copies of convolutional block

Data Exploration

Images – 32x32 pixels (smaller size will help the Neural Network train faster)

10 different types of classes

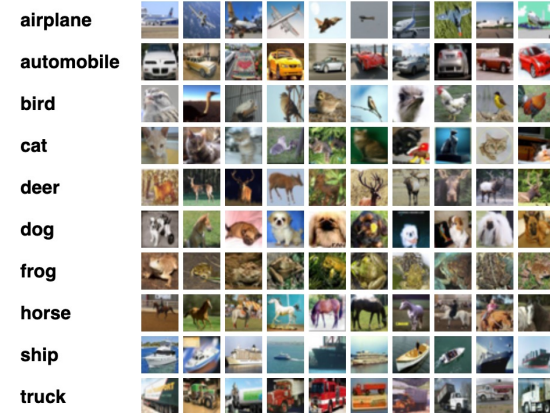
Total images - 60,000 . Training data : 50,000, Test data : 10,000

The CIFAR-10 dataset

The CIFAR-10 dataset consists of 60000 32x32 colour images in 10 classes, with 6000 images per class. There are 50000 training images and 10000 test images.

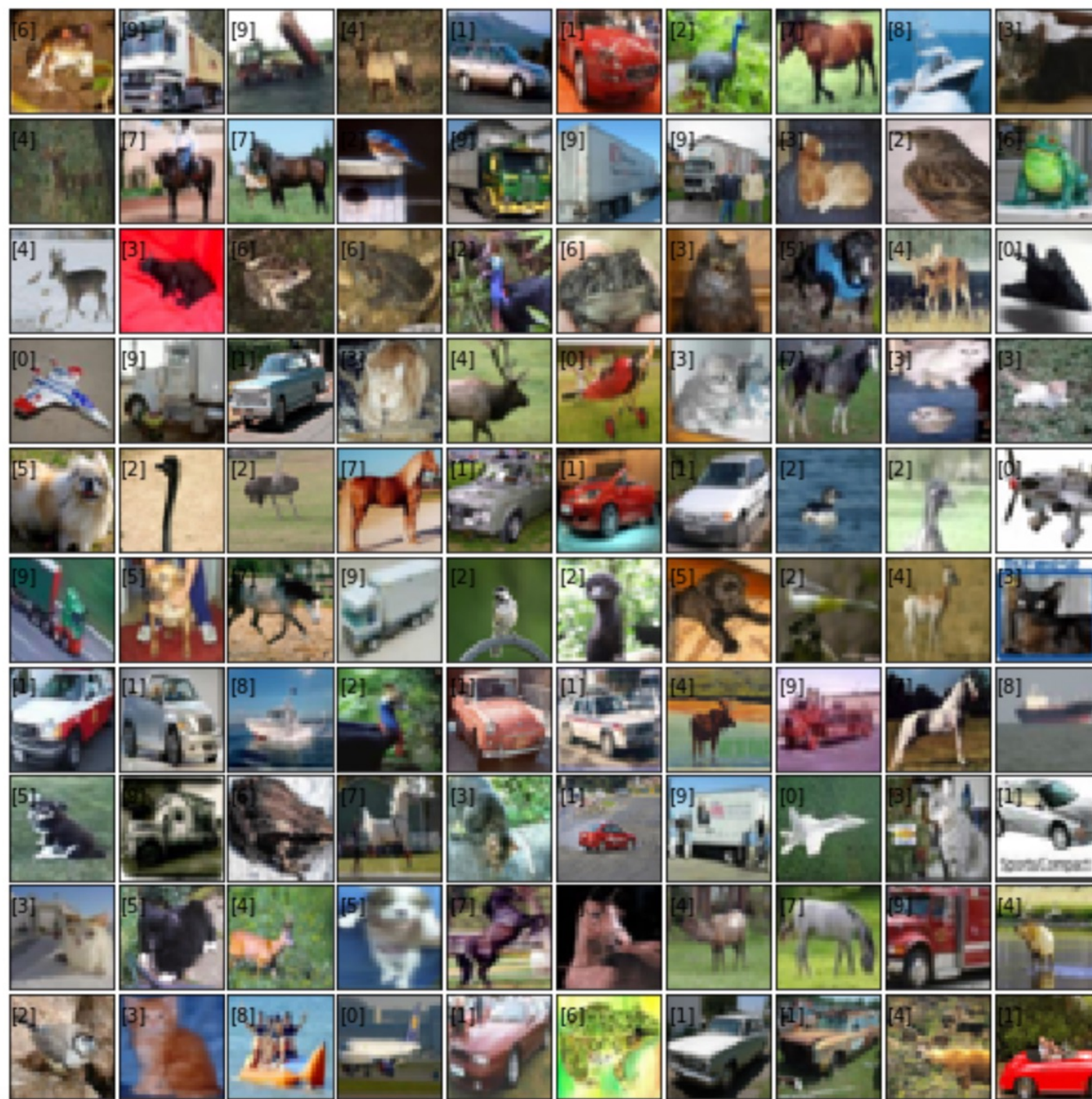
The dataset is divided into five training batches and one test batch, each with 10000 images. The test batch contains exactly 1000 randomly-selected images from each class. The training batches contain the remaining images in random order, but some training batches may contain more images from one class than another. Between them, the training batches contain exactly 5000 images from each class.

Here are the classes in the dataset, as well as 10 random images from each:



The classes are completely mutually exclusive. There is no overlap between automobiles and trucks. "Automobile" includes sedans, SUVs, things of that sort. "Truck" includes only big trucks. Neither includes pickup trucks.

0: 'Plane',
 1: 'Car',
 2: 'Bird',
 3: 'Cat',
 4: 'Deer',
 5: 'Dog',
 6: 'Frog',
 7: 'Horse',
 8: 'Boat',
 9: 'Truck'



Create a Neural Network from scratch

Steps :

Data Preprocessing :

- Load dataset and split data into train and test
- Convert data into float and normalize
- Convert each label (0 to 9) to an array of 10 digits with only one element set to 1 and rest set to 0's

Create a sequential model and add layers

- Conv2D, Conv2D,MaxPool2D,Dropout – 4 layers
- Conv2D, Conv2D,MaxPool2D,Dropout – 4 layers
- Flatten
- Dense layer
- Dropout
- Dense output layer

Compile model :

- loss = categorical cross entropy
- metric = accuracy

Train model :

- Epochs = 30, Batch size = 64
- Training time = 1298.42 secs/ 21 minutes
- Loss = .3013 , Accuracy =.8950

Save model structure and weights in files

Predict :

- Load trained model's structure and weights
- Use test image – convert to numpy array and normalize, add fourth dimension
- Predict

Model: "sequential"

Layer (type)	Output Shape	Param #
=====		
conv2d (Conv2D)	(None, 32, 32, 32)	896
conv2d_1 (Conv2D)	(None, 30, 30, 32)	9248
max_pooling2d (MaxPooling2D)	(None, 15, 15, 32)	0
dropout (Dropout)	(None, 15, 15, 32)	0
conv2d_2 (Conv2D)	(None, 15, 15, 64)	18496
conv2d_3 (Conv2D)	(None, 13, 13, 64)	36928
max_pooling2d_1 (MaxPooling2D)	(None, 6, 6, 64)	0
dropout_1 (Dropout)	(None, 6, 6, 64)	0
flatten (Flatten)	(None, 2304)	0
dense (Dense)	(None, 512)	1180160
dropout_2 (Dropout)	(None, 512)	0
dense_1 (Dense)	(None, 10)	5130
=====		

Total params: 1,250,858

Trainable params: 1,250,858

Non-trainable params: 0

- Number of parameters in max pool layer is 0. Instead of increasing the size of the Neural Network, they help us decrease the size by scaling down the data that passes through them, keeping the most important values.
This will help speed up the training process
- As we are just throwing data, dropout does not add any parameters to Neural Network

Custom Neural Network – trained on 50000 images, tested on 10000 images. Has 10 classes

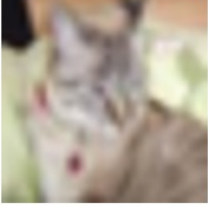







Image	Predicted class label	Predicted class likelihood	Result
	Cat	99.79	Correct
	Frog	96.32	Incorrect
	Cat	76.82	Incorrect
	Bird	80.62	Incorrect

Image	Predicted class label	Predicted class likelihood %	Result
	Frog	99.88	Correct
	Deer	70.84	Correct
	Plane	99.49	Incorrect
	Horse	99.84	Correct

Use Pre-trained models : VGG16





Dataset of millions of labeled pictures. It is a deep neural network that is 16-19 layers deep. Created by University of Oxford in 2014.

Trained on Imagenet database - <https://www.image-net.org/about.php>.

Steps :

- Create a VGG16 model
- Load test image
- Convert to numpy array, add fourth dimension
- Normalize data
- Make prediction
- Decode prediction

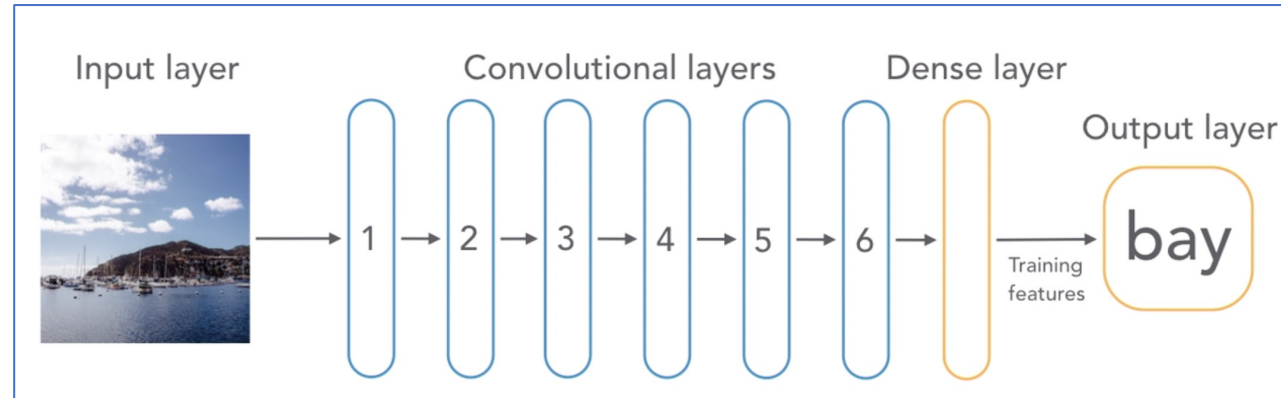
VGG has 16 layers. Trained on approximate 14 million images and has 1000 classes

Image	class label - probability of the input image being the class label	Image	class label - probability of the input image being the class label
	Prediction : grey_whale - 0.17140811681747437 Prediction : great_white_shark - 0.1249302476644516 <u>Prediction : catamaran - 0.12052340805530548</u> Prediction : tiger_shark - 0.10662741214036942 Prediction : submarine - 0.0976378321647644 Prediction : electric_ray - 0.05705215781927109 <u>Prediction : trimaran - 0.040302082896232605</u> Prediction : stingray - 0.037694282829761505 Prediction : wreck - 0.03490995615720749		Prediction : vulture - 0.36719706654548645 Prediction : white_stork - 0.212053120136261 Prediction : kite - 0.08006201684474945 Prediction : pelican - 0.07709106802940369 Prediction : bald_eagle - 0.05074508115649223 Prediction : black_stork - 0.04751254990696907 Prediction : goose - 0.02254302054643631 Prediction : parachute - 0.022074850276112556 Prediction : oystercatcher - 0.017053183168172836
	Prediction : mask - 0.33832019567489624 Prediction : wig - 0.3035263419151306 Prediction : macaw - 0.16652709245681763 Prediction : feather_boa - 0.06570449471473694 Prediction : hair_spray - 0.011579805985093117 Prediction : comic_book - 0.00953880324959755 Prediction : lorikeet - 0.00846063531935215 Prediction : jersey - 0.006699726916849613 Prediction : cloak - 0.005206170491874218		Prediction : Blenheim_spaniel - 0.7203316688537598 Prediction : Welsh_springer_spaniel - 0.11424615234136581 Prediction : Brittany_spaniel - 0.04804360866546631 Prediction : Saint_Bernard - 0.03310669586062431 <u>Prediction : cocker_spaniel - 0.02941340021789074</u> Prediction : English_springer - 0.013727271929383278 Prediction : Japanese_spaniel - 0.009968039579689503 Prediction : Sussex_spaniel - 0.009588563814759254 Prediction : basset - 0.004175855312496424

Transfer Learning

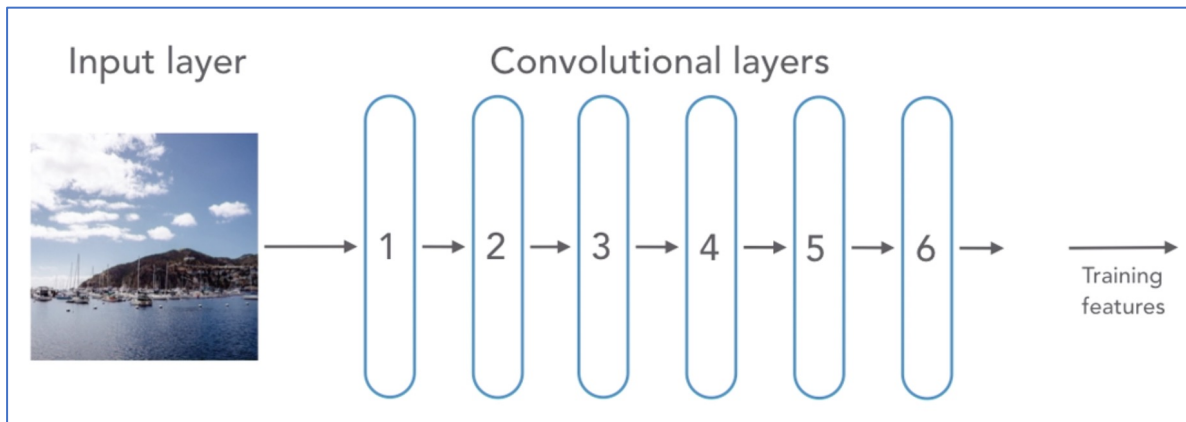
Use a model trained on one set of data as a starting point for modeling a new set of data

Standard Convolutional Neural Network

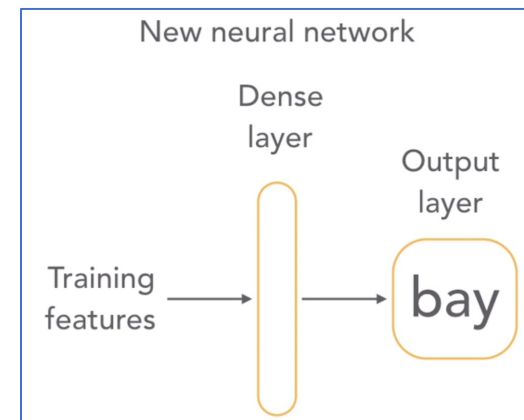


Transfer learning

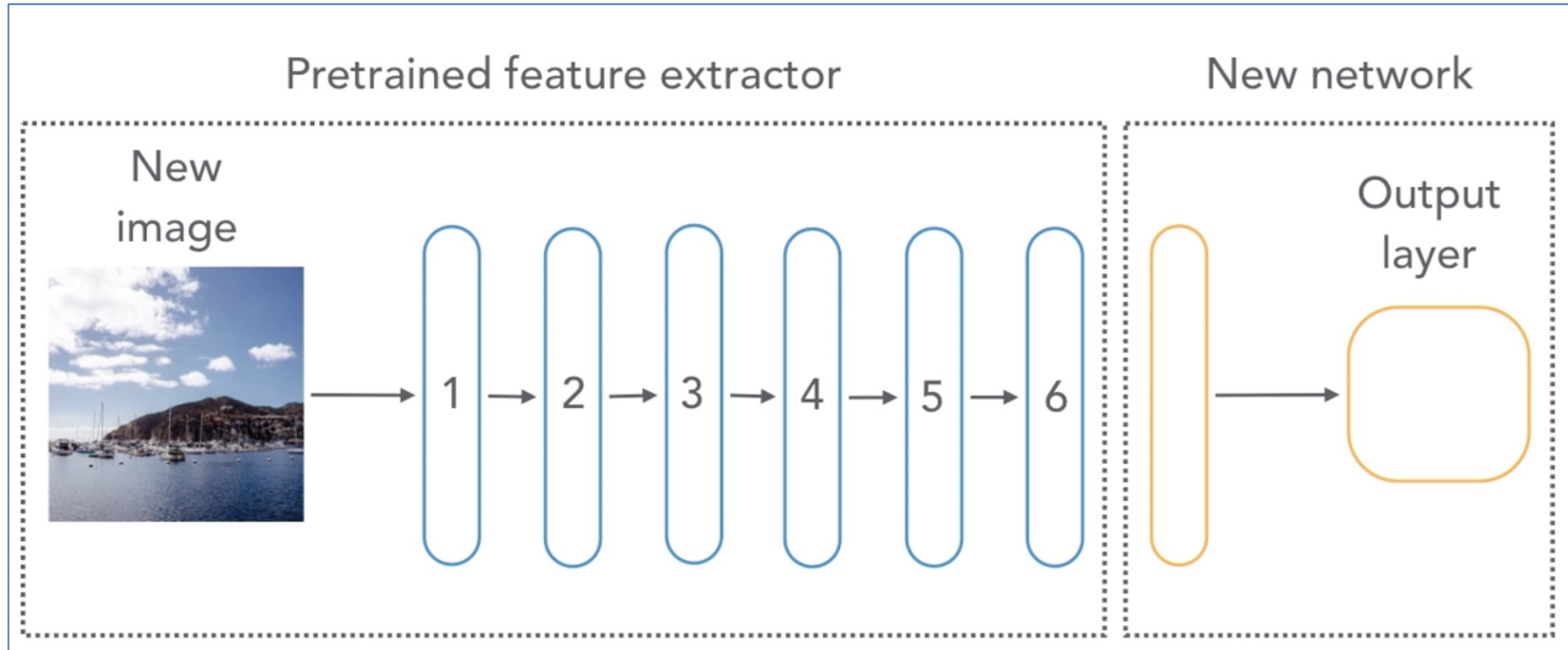
Feature Extractor - Remove the last layer



Create a new Neural Network to replace the last layer we sliced off



Predict with transfer learning



Steps :

Extract features with pre trained neural network:

- Create images list - 32 dog images and 32 not-dogs images from imagenet database
- Create labels list – 0 for dog and 1 for non-dog
- Converted lists to numpy arrays
- Normalize images array
- Use a pre-trained NN (VGG16) to be used as a feature extractor, removed the last/top layer
- Make prediction on normalized images array using above model.
- Save predictions and labels array







Train new NN with extracted features in the last step:

- Load extracted features (images + labels)
- Create a sequential model using only Flatten and Dense layers
- Compile and fit the model
- Save the model structure and weights

Make Prediction using new images

- Use saved model structure and weights
- Load test image, converted to numpy array, normalize it
- Extract features of test image using VGG16 and make prediction
- Make final prediction using saved model

Trained on 64 images from imagenet database – 32 dogs + 32 not dogs

Image	Percent likelihood this image contains a dog	Image	Percent likelihood this image contains a dog
	100		0
	100		100
	100		0

Conclusion

	Training Data	Training time	Accuracy of results
Create model from scratch	50000 images	1298 seconds	Some what Accurate
Pre trained model - VGG16	Million images	NA	More Accurate
Transfer Learning	64 images	2.45 seconds	Most Accurate

Thank You