Deep Convolutional Neural Networks

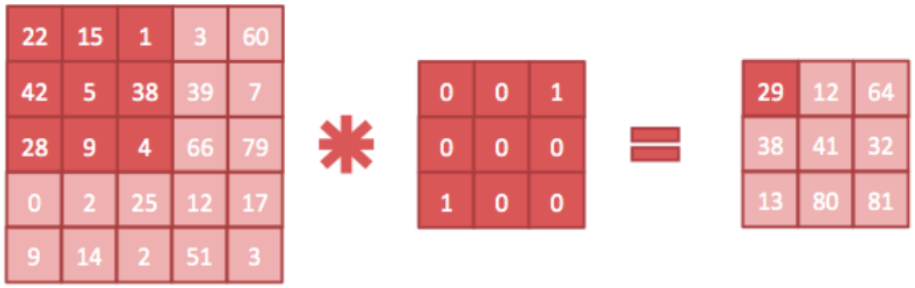
Dr. Fayyaz ul Amir Afsar Minhas
PIEAS Biomedical Informatics Research Lab
Department of Computer and Information Sciences
Pakistan Institute of Engineering & Applied Sciences
PO Nilore, Islamabad, Pakistan
http://faculty.pieas.edu.pk/fayyaz/

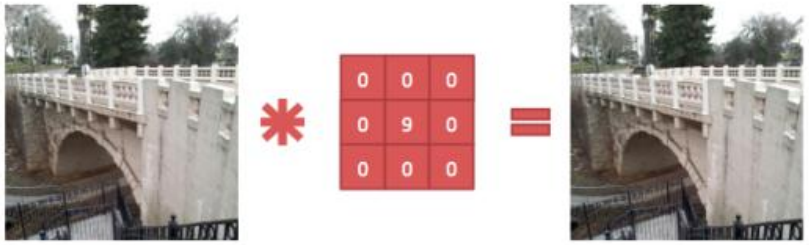
DCN: Deep Convolutional Networks

- A feed-forward network inspired from visual cortex
- Used for image recognition
- Objective
  - Find a set of filters which, when convolved with image, lead to the solution of the desired image recognition task
    - Invariant
    - Hierarchical
      - Increasing feature complexity
      - Increasing “Globality”

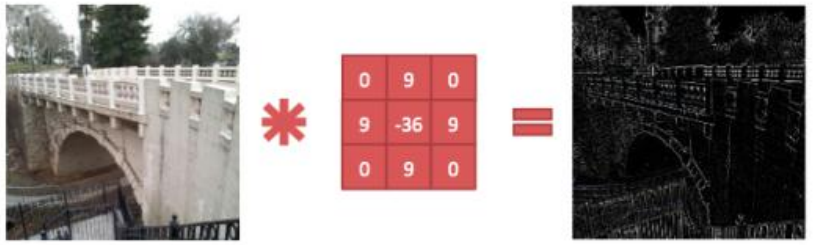
Basics

- The convolution operation

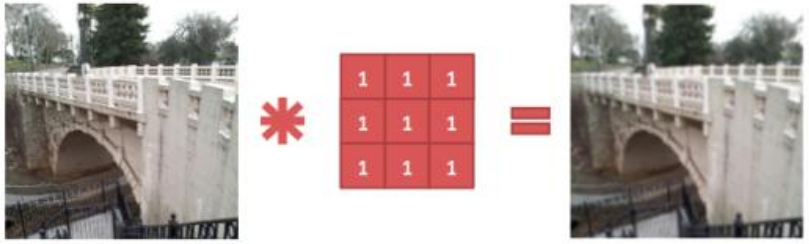




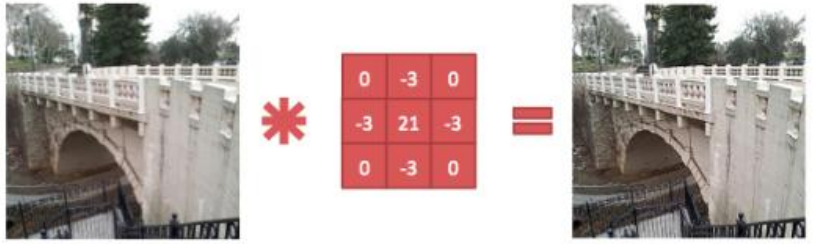
(a) Identity kernel



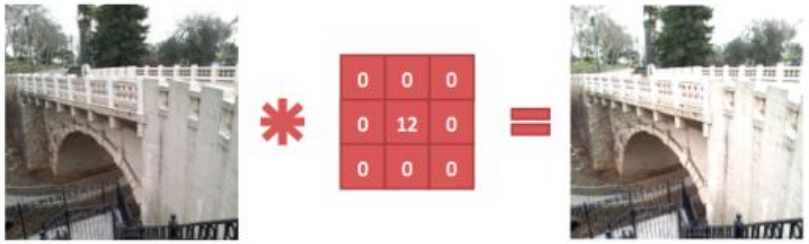
(b) Edge detection kernel



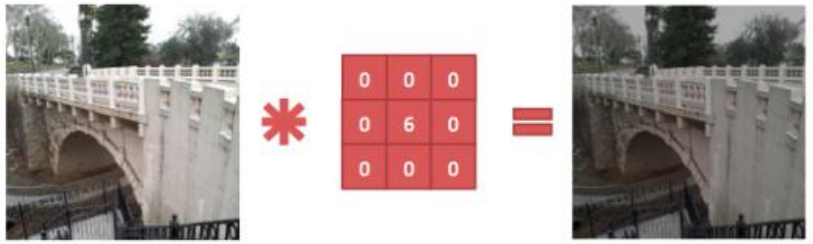
(c) Blur kernel



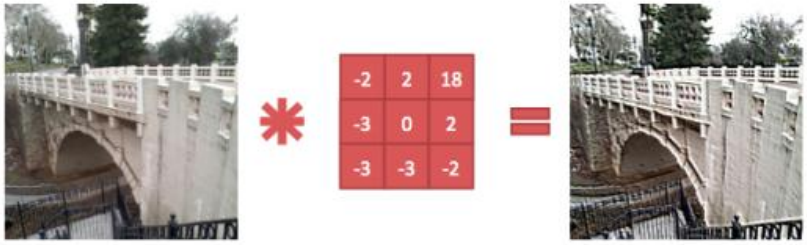
(d) Sharpen kernel



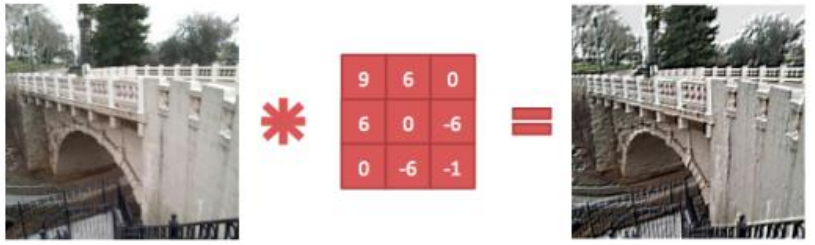
(e) Lighten kernel



(f) Darken kernel



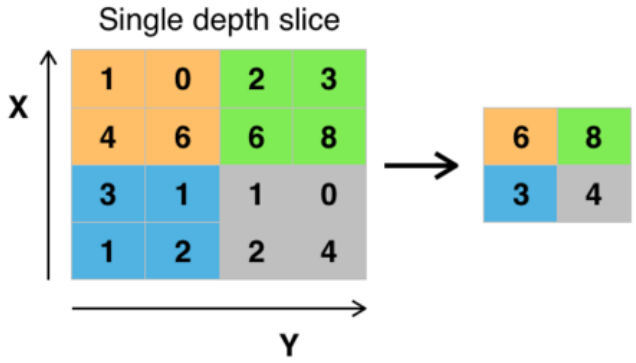
(g) Random kernel 1

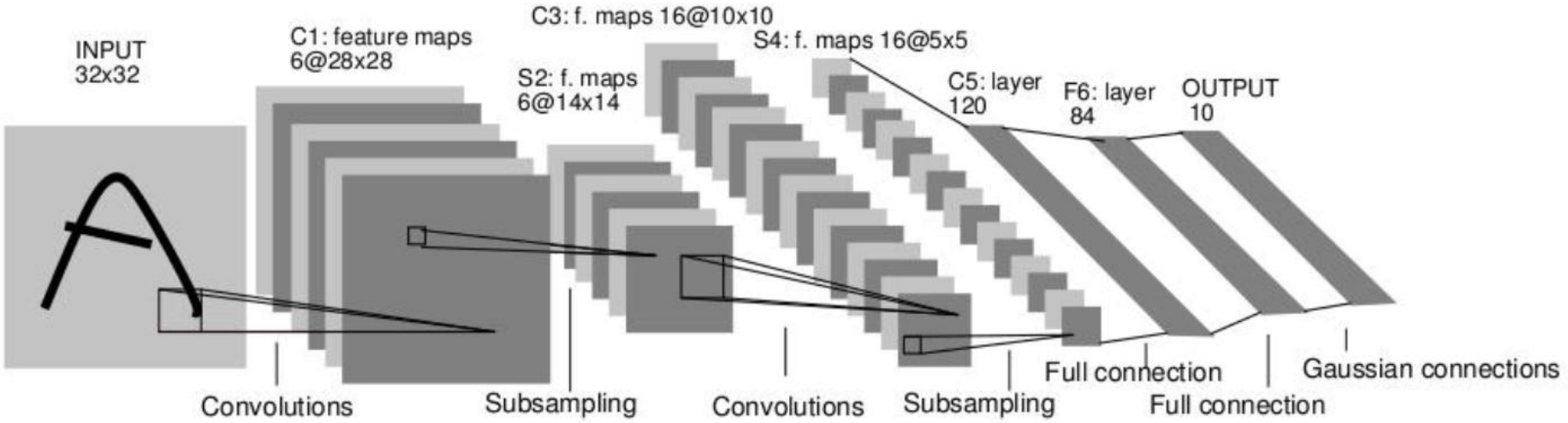


(h) Random kernel 2

Structure

- Increasing "globality"
  - Input → Convolution → Non-linearity → Sub-sampling … → Fully Connected Layer (for classification)





```
import numpy
from keras.datasets import cifar10
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import Dropout
from keras.layers import Flatten
from keras.constraints import maxnorm
from keras.optimizers import SGD
from keras.layers.convolutional import Conv2D
from keras.layers.convolutional import MaxPooling2D
from keras.utils import np_utils
from keras import backend as K
K.set_image_dim_ordering('th')
# fix random seed for reproducibility
seed = 7
numpy.random.seed(seed)
# load data
(X_train, y_train), (X_test, y_test) = cifar10.load_data()
X_train = X_train.astype('float32')
X_test = X_test.astype('float32')
X_train = X_train / 255.0
X_test = X_test / 255.0
y_train = np_utils.to_categorical(y_train)
y_test = np_utils.to_categorical(y_test)
num_classes = y_test.shape[1]
```

http://machinelearningmastery.com/object-recognition-convolutional-neural-networks-keras-deep-learning-library/

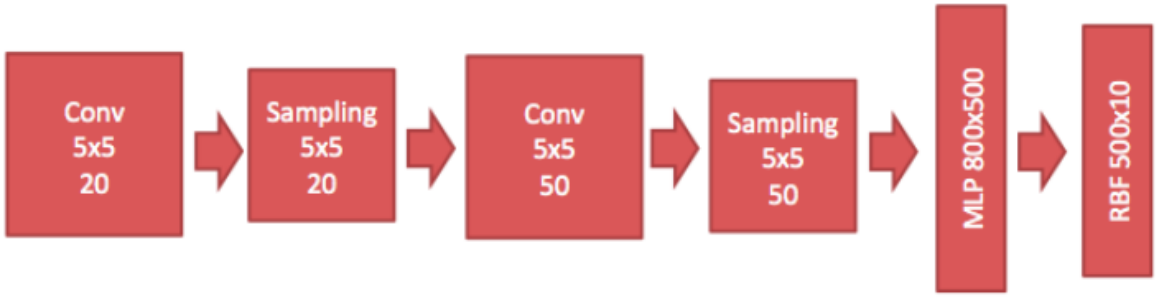
```
# Create the model
model = Sequential()
model.add(Conv2D(32, (3, 3), input_shape=(3, 32, 32), padding='same', activation='relu',
kernel_constraint=maxnorm(3)))
model.add(Dropout(0.2))
model.add(Conv2D(32, (3, 3), activation='relu', padding='same', kernel_constraint=maxnorm(3)))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Flatten())
model.add(Dense(512, activation='relu', kernel_constraint=maxnorm(3)))
model.add(Dropout(0.5))
model.add(Dense(num_classes, activation='softmax'))

# Compile model
epochs = 25
lrate = 0.01
decay = lrate/epochs
sgd = SGD(lr=lrate, momentum=0.9, decay=decay, nesterov=False)
model.compile(loss='categorical_crossentropy', optimizer=sgd, metrics=['accuracy'])
print(model.summary())

# Fit the model
model.fit(X_train, y_train, validation_data=(X_test, y_test), epochs=epochs, batch_size=32)
# Final evaluation of the model
scores = model.evaluate(X_test, y_test, verbose=0)
print("Accuracy: %.2f%%" % (scores[1]*100))
```

Layer (type)	Output Shape	Param #
conv2d_1 (Conv2D)	(None, 32, 32, 32)	896
dropout_1 (Dropout)	(None, 32, 32, 32)	0
conv2d_2 (Conv2D)	(None, 32, 32, 32)	9248
max_pooling2d_1 (MaxPooling2	(None, 32, 16, 16)	0
flatten_1 (Flatten)	(None, 8192)	0
dense_1 (Dense)	(None, 512)	4194816
dropout_2 (Dropout)	(None, 512)	0
dense_2 (Dense)	(None, 10)	5130

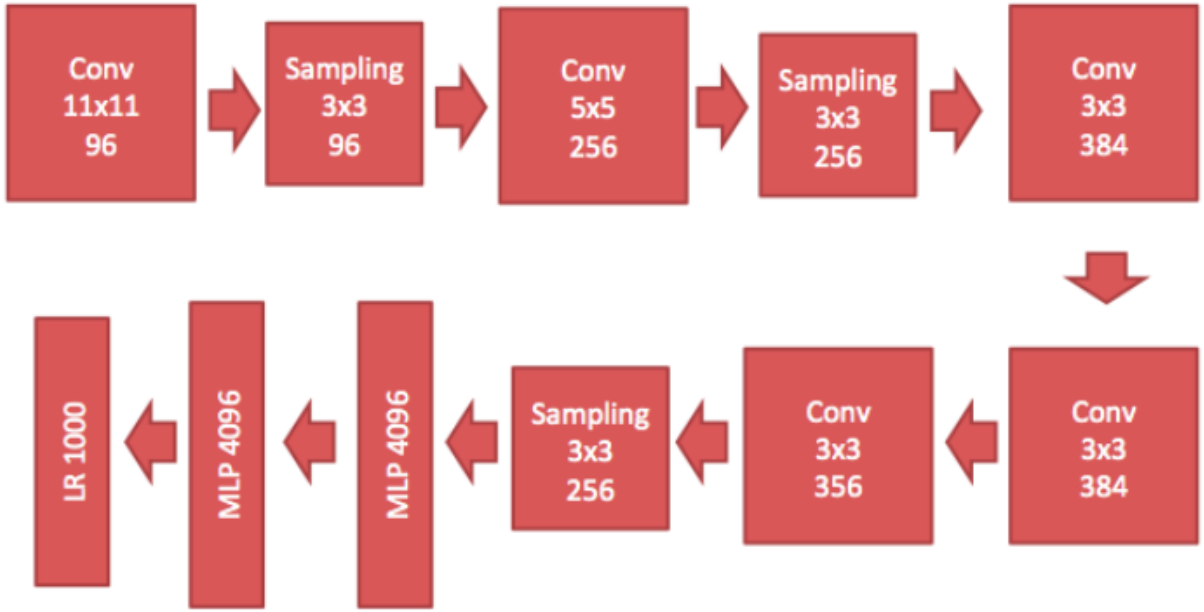
Total params: 4,210,090.0
Trainable params: 4,210,090.0
Non-trainable params: 0.0

Famous CNN

- LeNet (Le Cunn 1990, 1998)



- AlexNet



Famous CNN

- VGG19
- Inception
- XCeption

Important Concepts

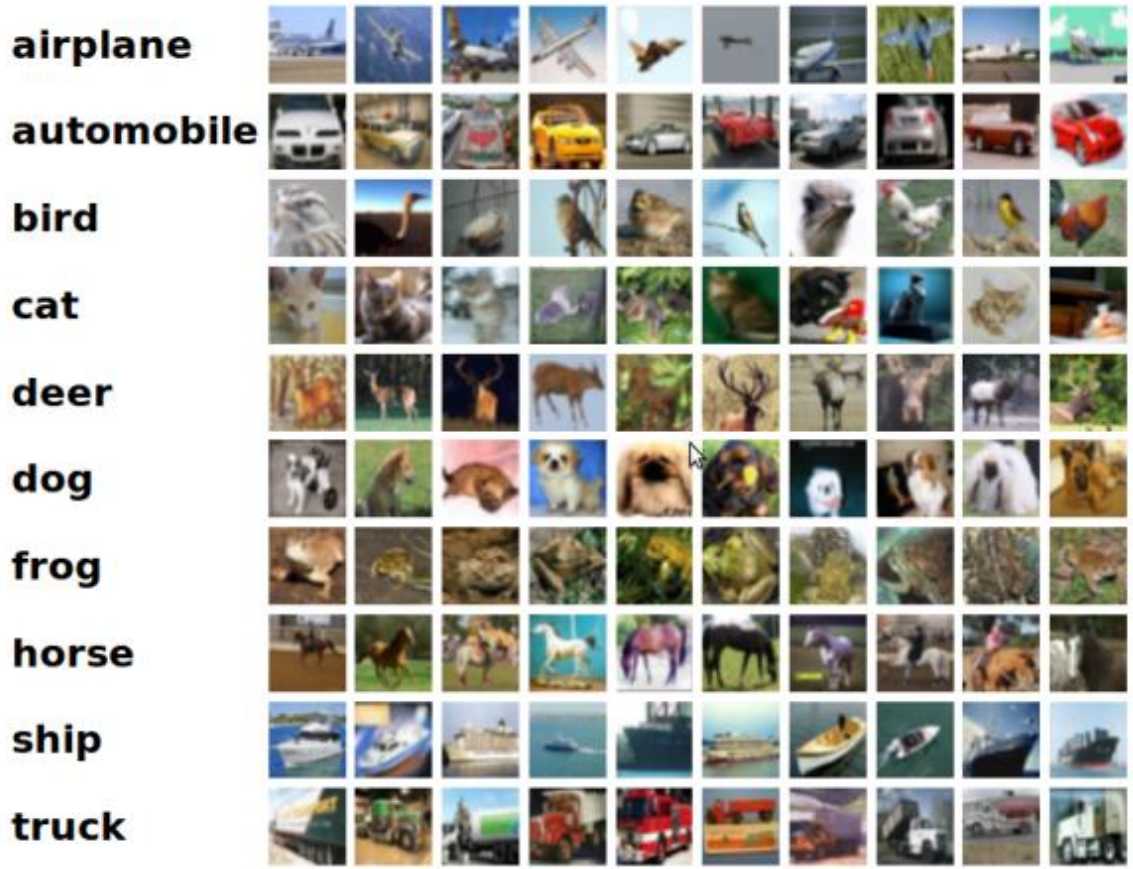
- Differences from fully connected nets
  - 3D volume of neurons
  - Local connectivity
  - Shared weights
- Hyper-parameter
  - Number of filters
  - Filter shape (receptive field)
  - Pooling type and shape
  - Regularization
    - Dropout
    - Data Augmentation
    - Early Stopping
    - Norm constraints
    - L1/L2 regularization

Important Concepts

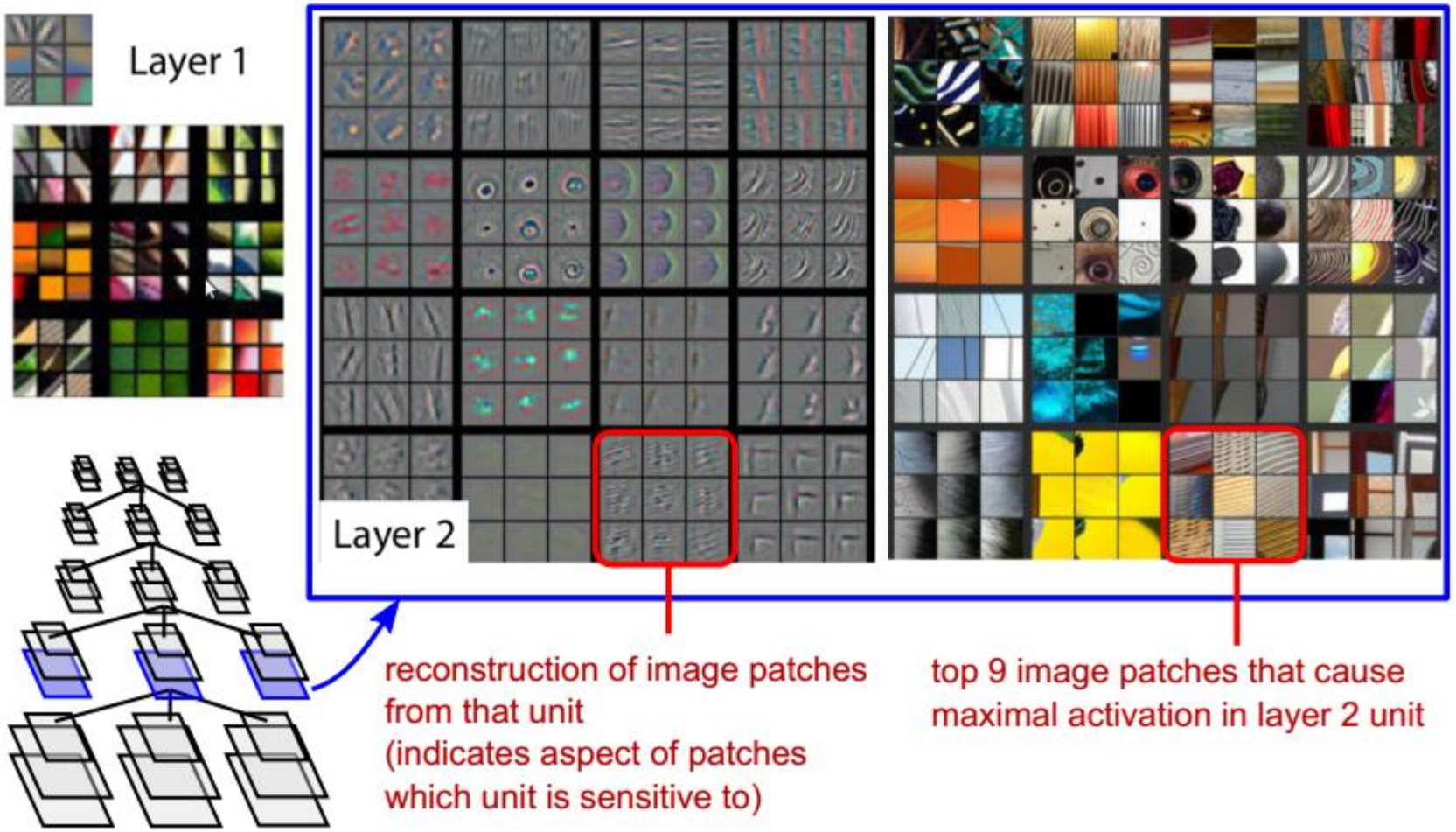
- Required reading: **ImageNet Classification with Deep Convolutional Neural Networks**
-

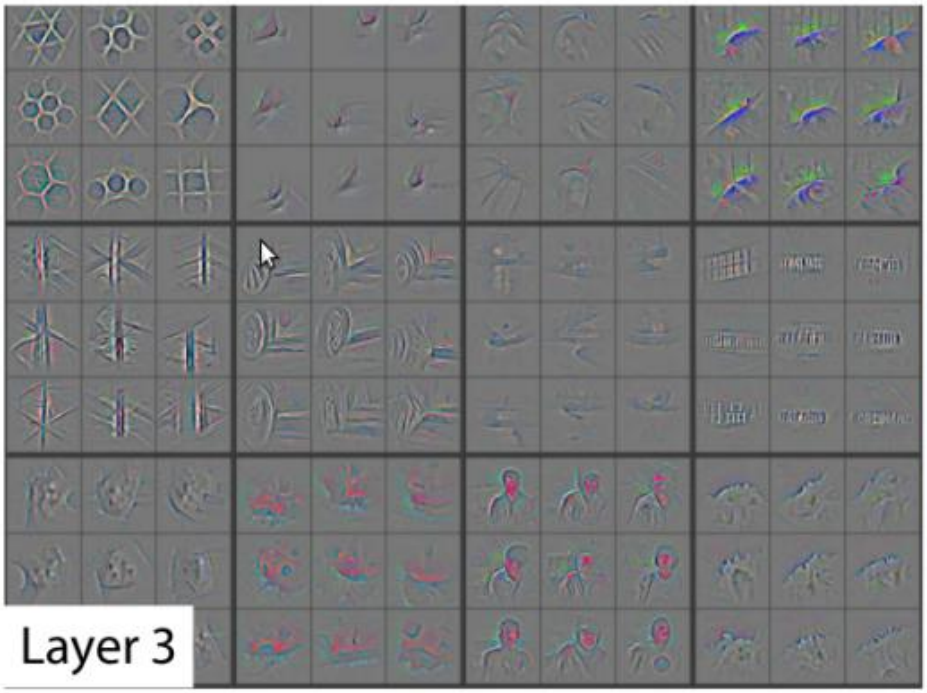
Visalization of a CNN

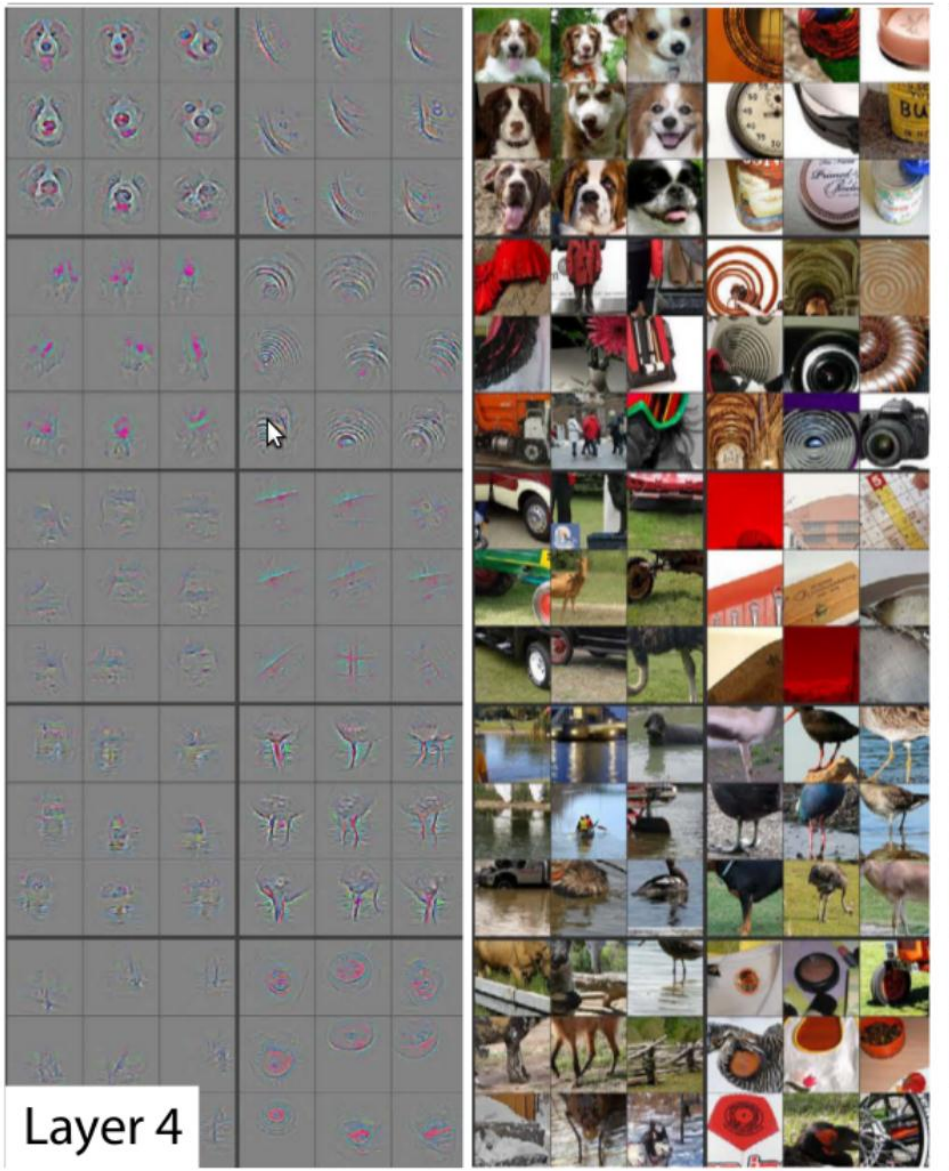
- **Visualizing and Understanding Convolutional Networks by Zeiler and Fergus, ECCV 2014, Part I, LNCS 8689, pp. 818–833, 2014.**

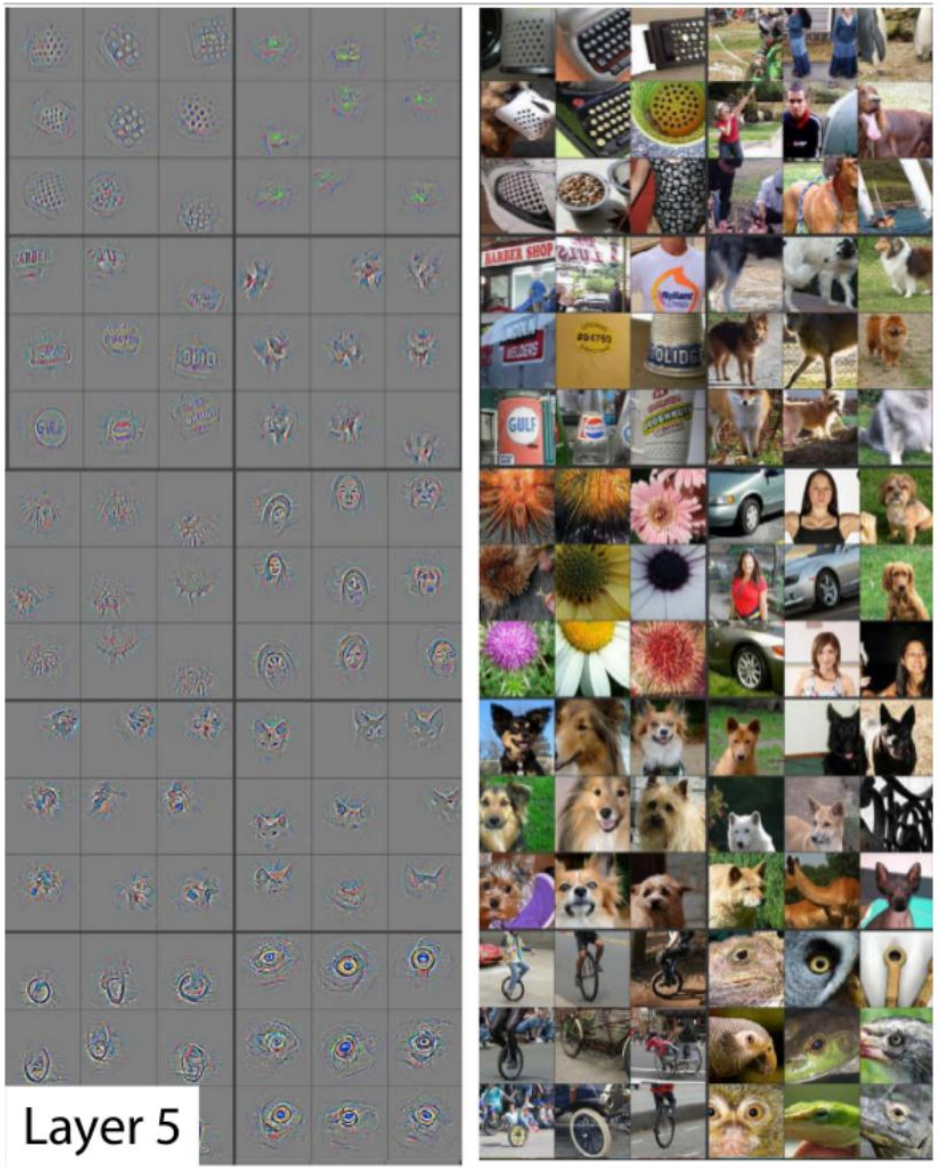


Visualization: https://blog.keras.io/how-convolutional-neural-networks-see-the-world.html


Layer 1
Layer 2
reconstruction of image patches from that unit (indicates aspect of patches which unit is sensitive to)
top 9 image patches that cause maximal activation in layer 2 unit


Layer 3


Layer 4


Layer 5

Fine Tuning/Transfer Learning

- A form of transfer learning
  - Use a pre-trained neural network for another classification task
    - Modify the weights of the final layers
- Trained image classification models for Keras
  - https://github.com/fchollet/deep-learning-models
- Transfer Learning: Recognition of traffic light
  - (https://medium.freecodecamp.com/recognizing-traffic-lights-with-deep-learning-23dae23287cc)
- Building powerful image classification models using very little data
  - https://blog.keras.io/building-powerful-image-classification-models-using-very-little-data.html

End of Lecture

We want to make a machine that will be proud of us.

- Danny Hillis