Deep Learning Techniques

TA. Hwanmoo Yong

Machine Learning &
Control Systems Lab.

Contents

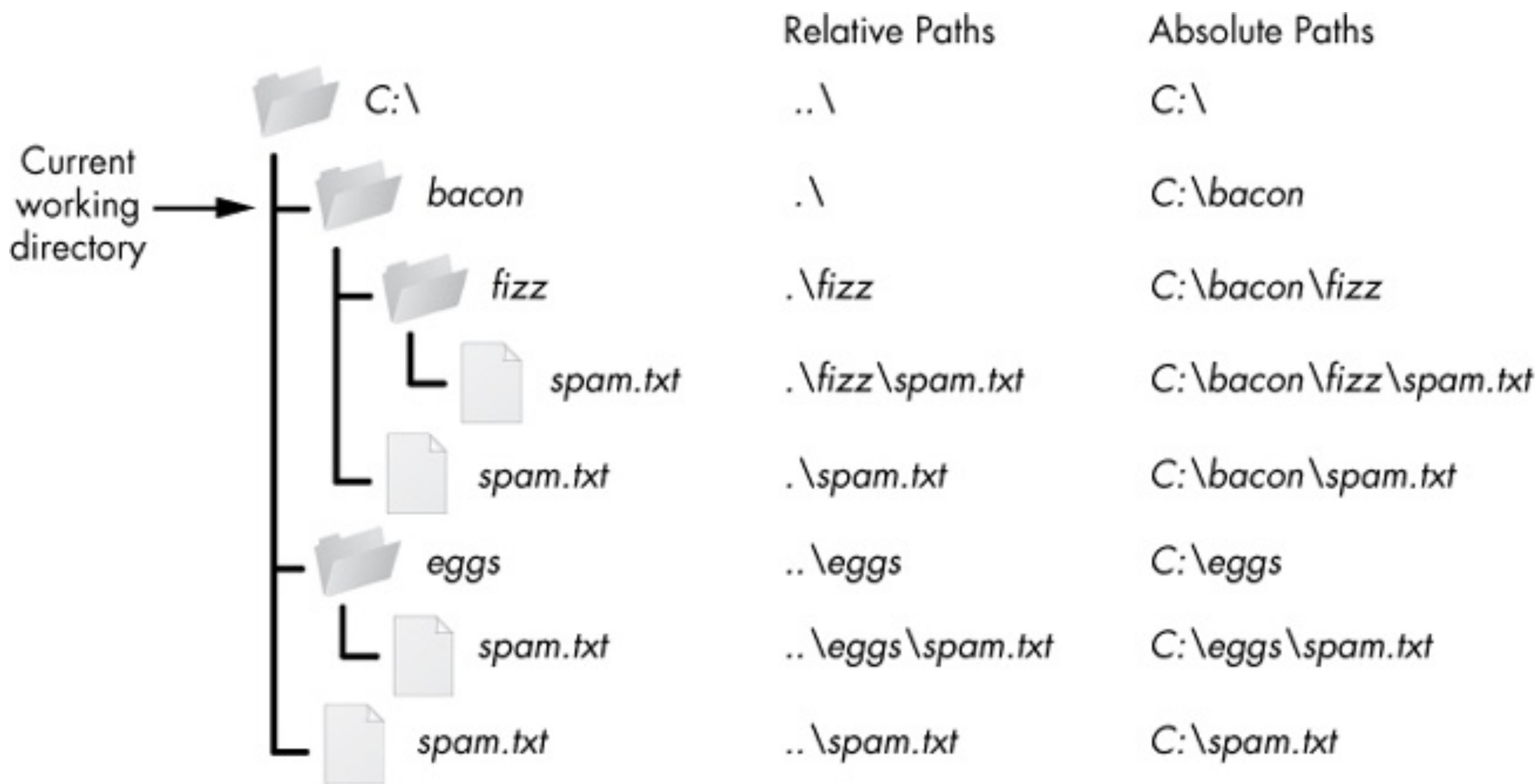
Custom Data Loader
- Relative, Absolute Paths
- Loading Files

Transfer Learning
- Convolutional Filters
- ImageNet
- VGG, ResNet, Inception

Custom Neural Networks using Keras
- Multiple Inputs and Outputs
- Using Transfer Learning

Data Augmentation
- Common Image Processing Techniques
- Some experimental techniques

Relative, Abosolute Paths



- It is preferred to use relative paths in most cases
- Use 'os.path' library instead of using string path

Loading Files

```
hmd_emotion — ssh lab.hwanmoo.kr — 98×18
Python 3.6.5 (default, Apr 25 2019, 14:52:32)
[GCC 5.4.0 20160609] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import os, glob
>>> data_path = os.path.join('./data','RafD')
>>> data_path
'./data/RafD'
>>> data = glob.glob(os.path.join(data_path,'*.jpg'))
>>> data
['./data/RafD/Rafd135_07_Caucasian_male_neutral_left.jpg', './data/RafD/Rafd045_22_Caucasian_femal
e_sad_left.jpg', './data/RafD/Rafd045_71_Caucasian_male_disgusted_right.jpg', './data/RafD/Rafd045
_02_Caucasian_female_neutral_right.jpg', './data/RafD/Rafd000_20_Caucasian_male_fearful_left.jpg',
 './data/RafD/Rafd045_42_Kid_male_sad_frontal.jpg', './data/RafD/Rafd090_69_Moroccan_male_sad_righ
t.jpg', './data/RafD/Rafd180_73_Moroccan_male_sad_frontal.jpg', './data/RafD/Rafd180_28_Caucasian_
male_disgusted_right.jpg', './data/RafD/Rafd180_28_Caucasian_male_contemptuous_frontal.jpg', './da
ta/RafD/Rafd000_40_Kid_male_disgusted_right.jpg', './data/RafD/Rafd090_65_Kid_female_angry_frontal
.jpg', './data/RafD/Rafd135_03_Caucasian_male_contemptuous_right.jpg', './data/RafD/Rafd045_03_Cau
casian_male_surprised_left.jpg', './data/RafD/Rafd090_45_Moroccan_male_angry_frontal.jpg', './data
```

- ‘os.path.join’ creates path.
- ‘glob.glob’ returns the files in given path, but note that the files are not sorted.

Loading Files (Images)


```python
import os, glob
import numpy as np

from PIL import Image

data_dir = os.path.join('./data','RafD')
image_paths = sorted(glob.glob(os.path.join(data_dir,'*.jpg')))

images = []
for image_index, image_path in enumerate(image_paths):
    image = Image.open(image_path)
    image_array = np.asarray(image)

    if image_index == 0:
        print("Reading image from ", image_path)

        print(image, type(image))
        print(image_array.shape, type(image_array))

    images.append(image_array)

images = np.asarray(images)
# Now you have all your images as numpy array

```

Loading Files (Images and labels)

```python
data_dir = os.path.join('./data','RafD')
image_paths = sorted(glob.glob(os.path.join(data_dir,'*.jpg')))[0:10]

EMOTIONS = ['happy','angry','sad','contemptuous','disgusted','neutral','fearful','surprised']

images = []
labels = []
for image_index, image_path in enumerate(image_paths):
    image = Image.open(image_path)
    image_array = np.asarray(image)

    image_name = os.path.split(image_path)[1]
    image_emotion = image_name.split('_')[4]

    image_label = EMOTIONS.index(image_emotion)

    if image_index == 0:
        print("Reading image from ", image_path)

        print(image, type(image))
        print(image_array.shape, type(image_array))

    images.append(image_array)
    labels.append(image_label)

images = np.asarray(images)
labels = np.asarray(labels)
# Now you have all your images and labels as numpy array

save_path = os.path.join(data_dir, 'data.npz')
np.savez(save_path, image=images, label=labels)

data = np.load(save_path)
images = data['image']
labels = data['label']
```

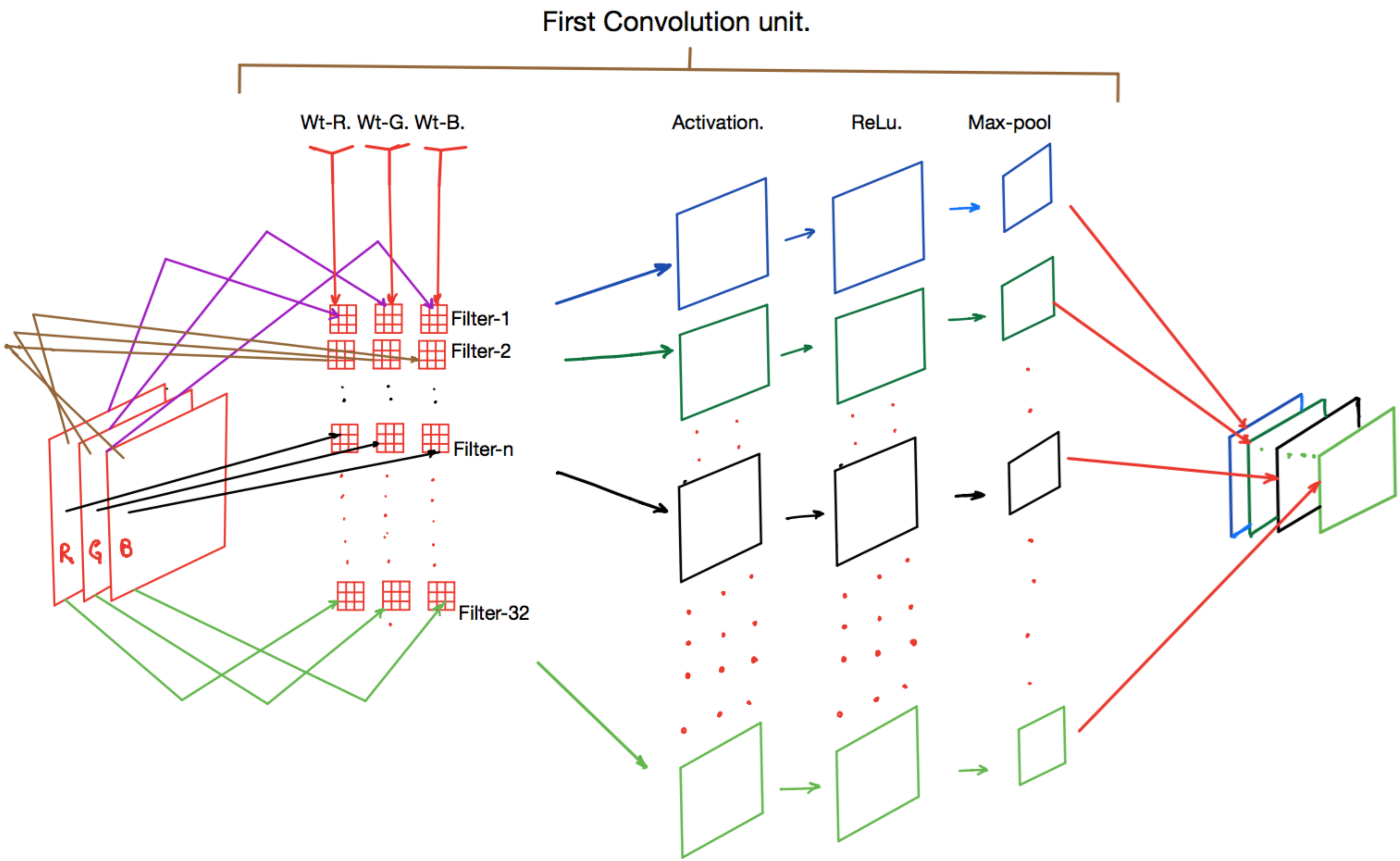
- We read the label from the file name.
- Saving data files into '.npz' file saves time when loading the dataset once you create it.

Loading Files (Using Yield)

```python
def data_generator(images, labels, cnt):
    i = 0
    while True:
        if i+cnt > len(labels):
            x = np.concatenate([images[i:len(images)],images[0:i+cnt-len(images)]],axis=0)
            y = np.concatenate([labels[i:len(labels)],labels[0:i+cnt-len(labels)]],axis=0)

            i = i+cnt-len(labels)
        else:
            x = images[i:i+cnt]
            y = labels[i:i+cnt]

            i = i+cnt

        yield x, y

data_gen = data_generator(images, labels, 1022)
x, y = next(data_gen)
print(y)

x, y = next(data_gen)
print(y)
```

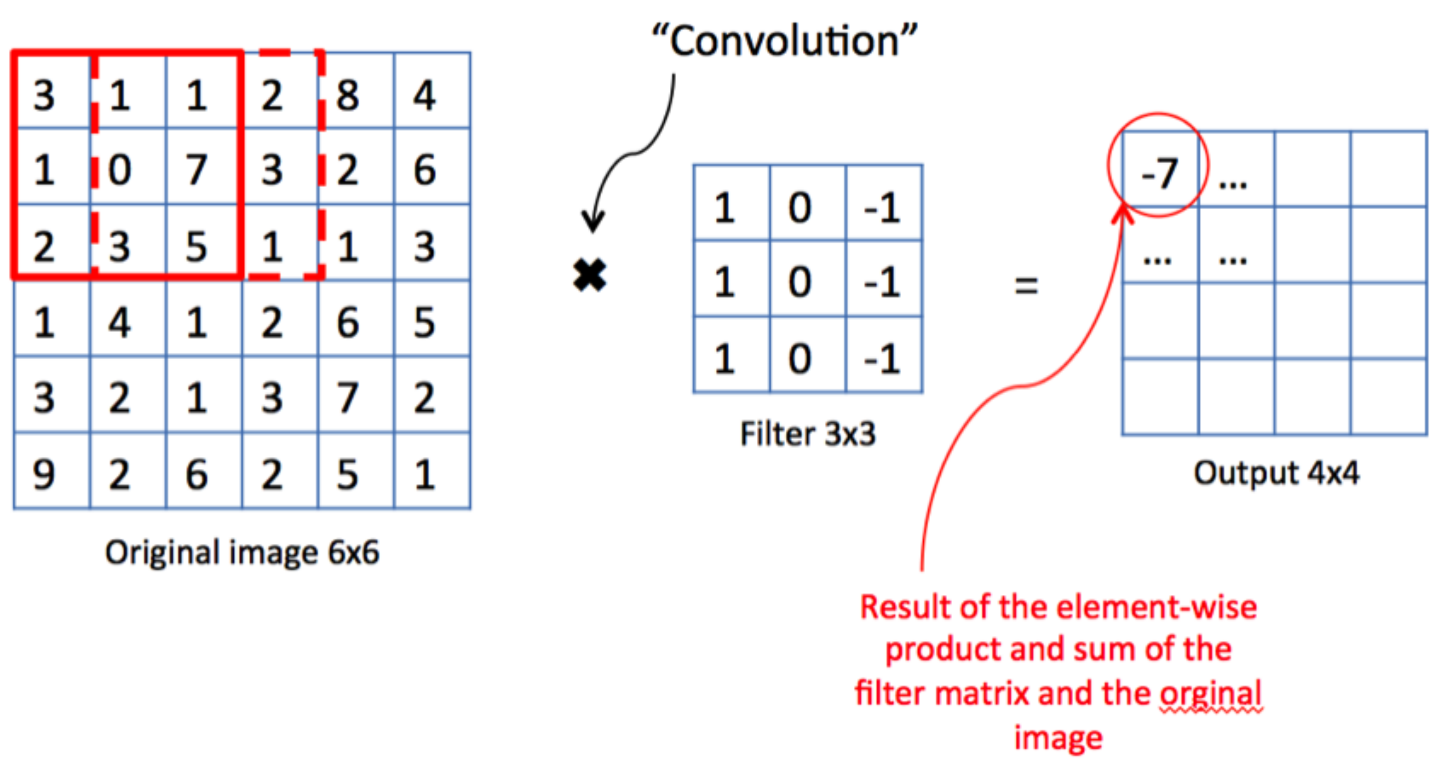
– Generator waits until you call it with ‘next()’ and it returns the value.

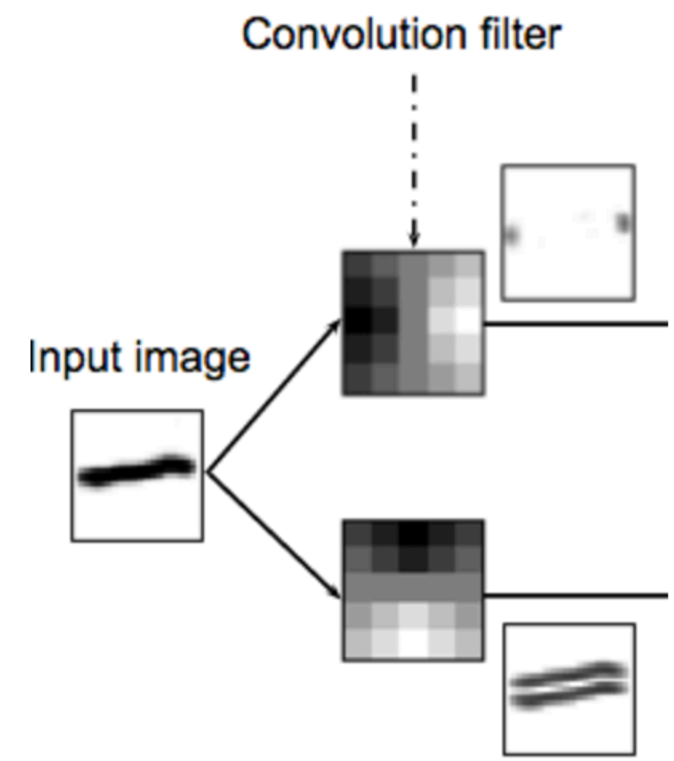
Convolution Layer

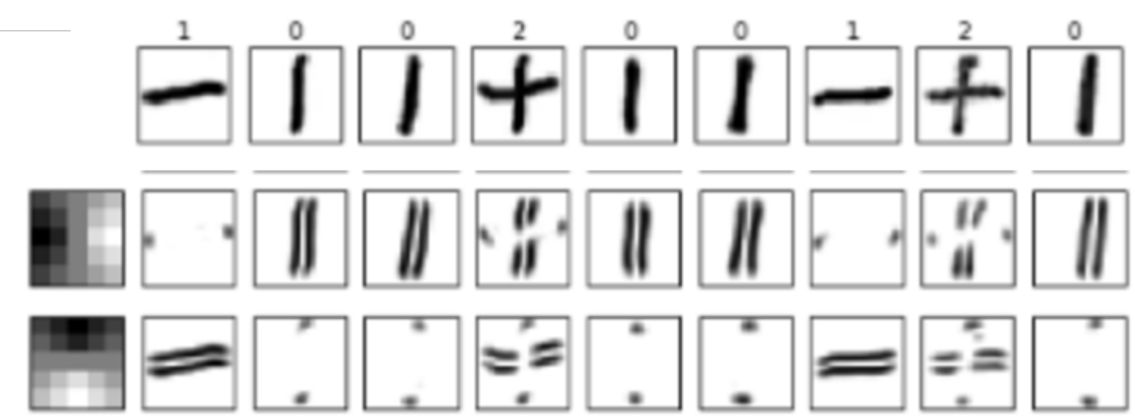


- The weights and the biases which are the components of the filters change when training the network.

Filters in CNN







- Note that the size of the output may be different from the size of the input.
- The output of the filter will be large if the input has the feature that the filter can detect.

ImageNet

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., ... & Fei-Fei, L. (2015). Imagenet large scale visual recognition challenge. *arXiv preprint arXiv:1409.0575.* [web]

14,197,122 images, 21841 synsets indexed

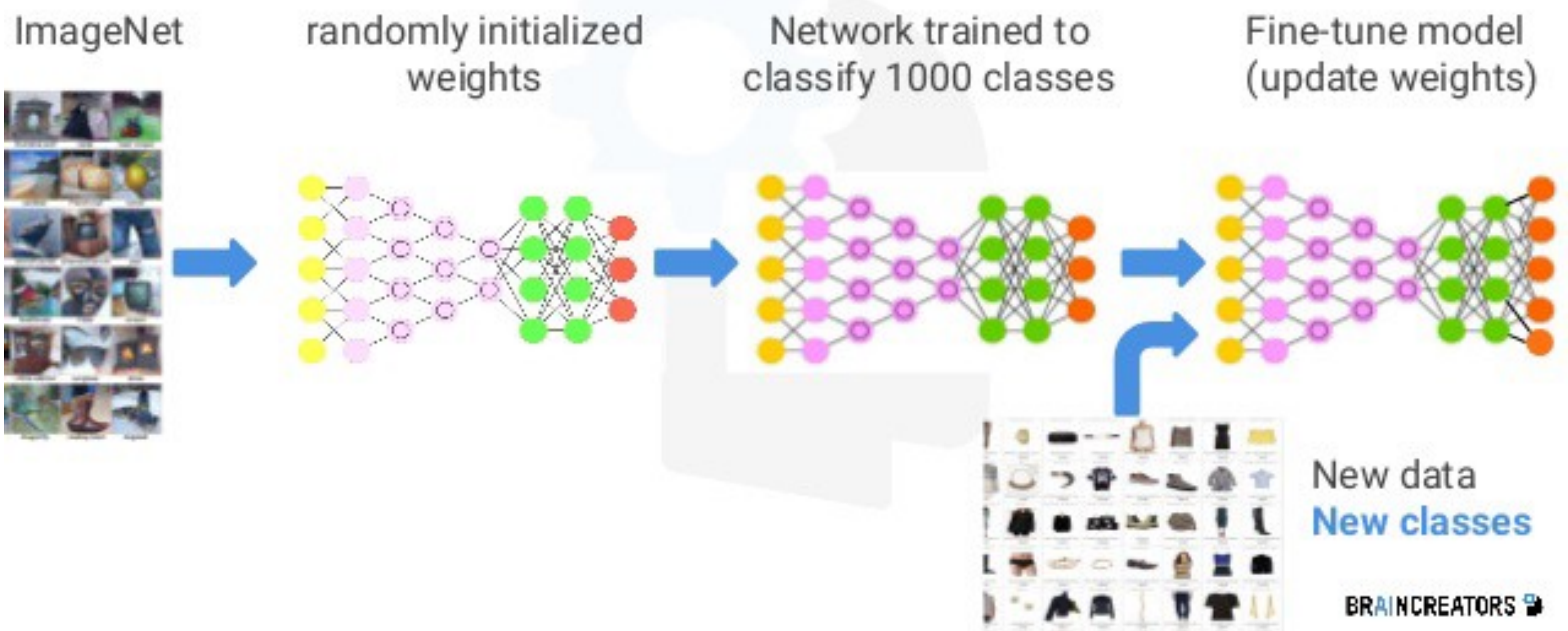
Explore Download Challenges Publications Updates About

Not logged in. Login I Signup

ImageNet is an image database organized according to the WordNet hierarchy (currently only the nouns), in which each node of the hierarchy is depicted by hundreds and thousands of images. Currently we have an average of over five hundred images per node. We hope ImageNet will become a useful resource for researchers, educators, students and all of you who share our passion for pictures.

Click here to learn more about ImageNet, Click here to join the ImageNet mailing list.

Transfer Learning



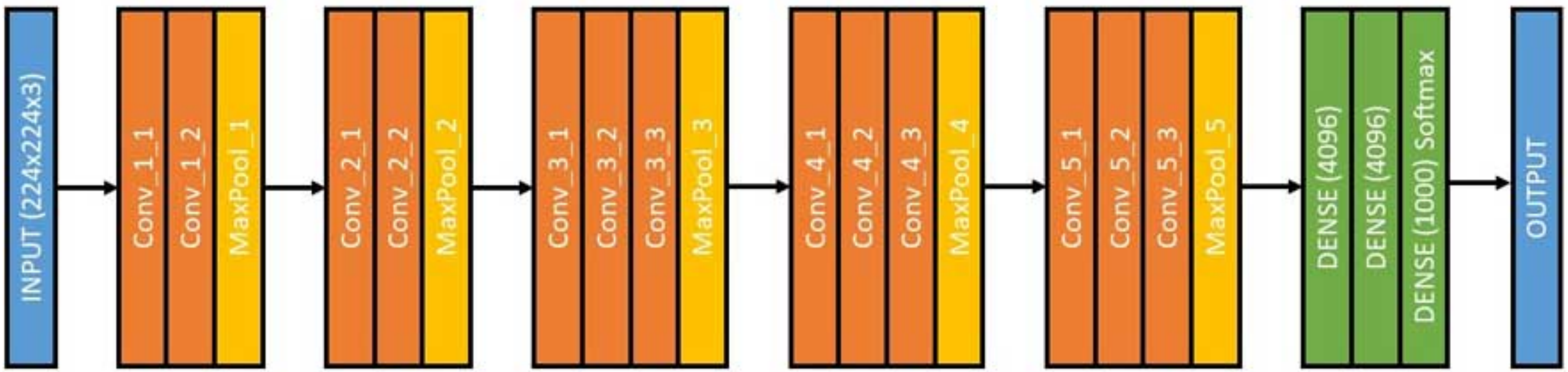
– The core idea of transfer learning is to get better initial weights to extract good visual features.

Transfer Learning using Keras

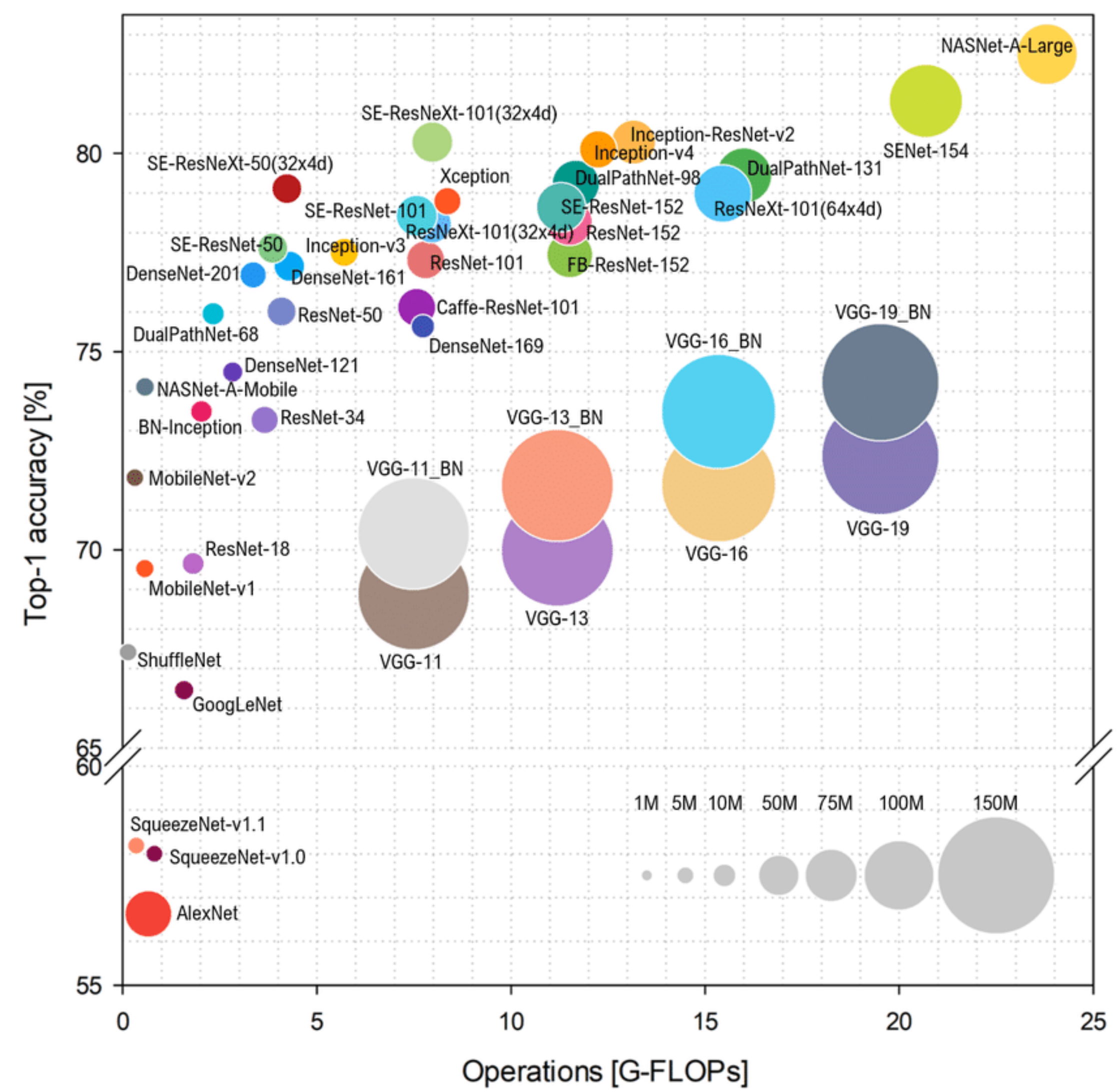
```python
base_model = keras.applications.densenet.DenseNet121(include_top=False,
                                                     weights='imagenet',
                                                     input_tensor=None,
                                                     input_shape=image_dim,
                                                     pooling='avg',
                                                     classes=nb_classes)
# base_model = keras.applications.inception_resnet_v2.InceptionResNetV2(include
# base_model = keras.applications.resnet50.ResNet50(include_top=False, weights=

x = Dense(256, activation='relu')(base_model.output)
x = Dropout(0.5)(x)
x = Dense(64)(x)

predictions = Dense(8, activation='softmax', name='fc1000')(x)

model = Model(input=base_model.input, output=predictions)

model_json = model.to_json()
with open("dense.json", "w") as json_file :
    json_file.write(model_json)
```

- include_top: whether to include the fully-connected layer at the top of the network.
- weights: one of None (random initialization), 'imagenet' (pre-training on ImageNet), or the path to the weights file to be loaded.

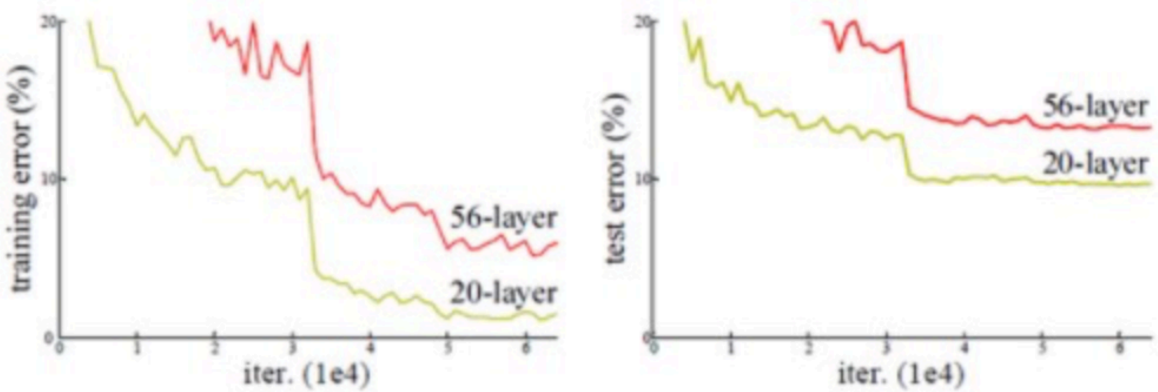
VGG

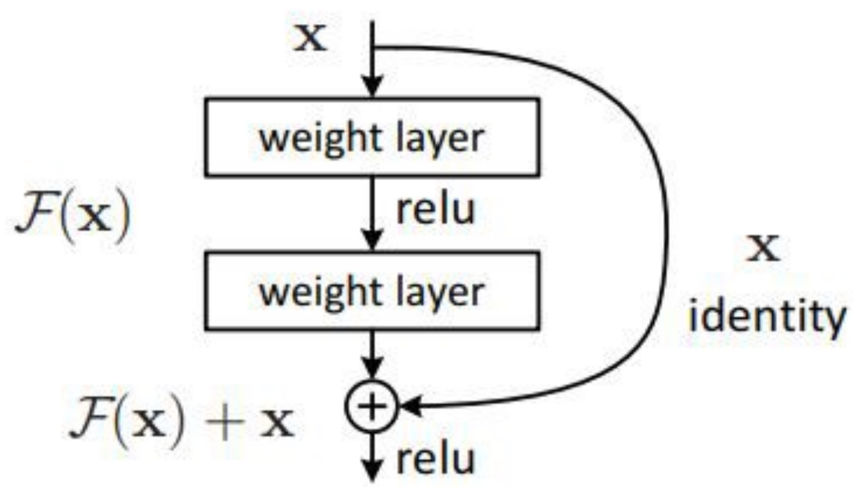


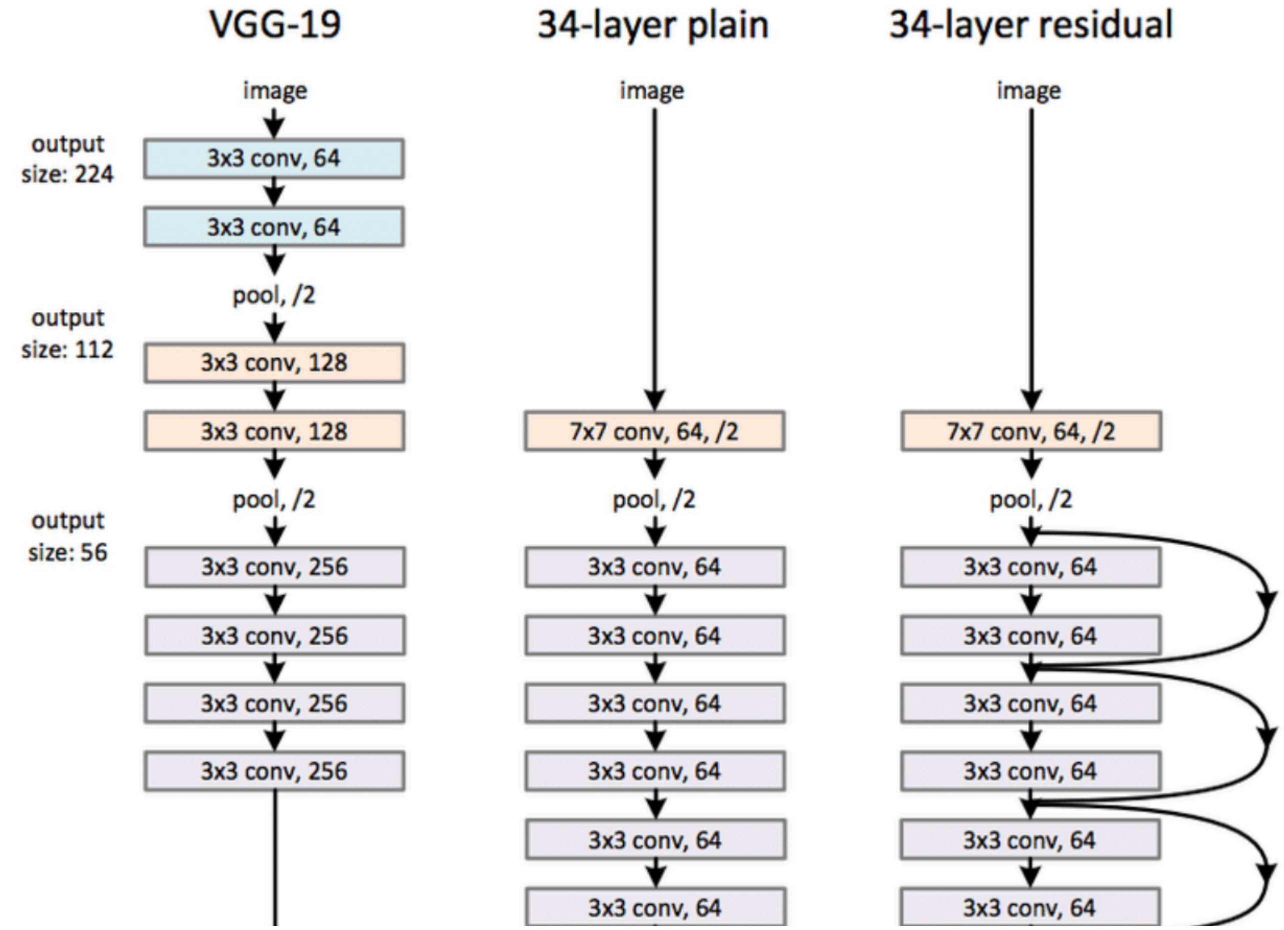
- VGG has quite simple structure, but it has too many parameters (because of the fully-connected layers).
- Using smaller but many kernels performs better because they have more non-linear functions.
- Selecting proper neural network is important especially when using it in embedded systems.



ResNet





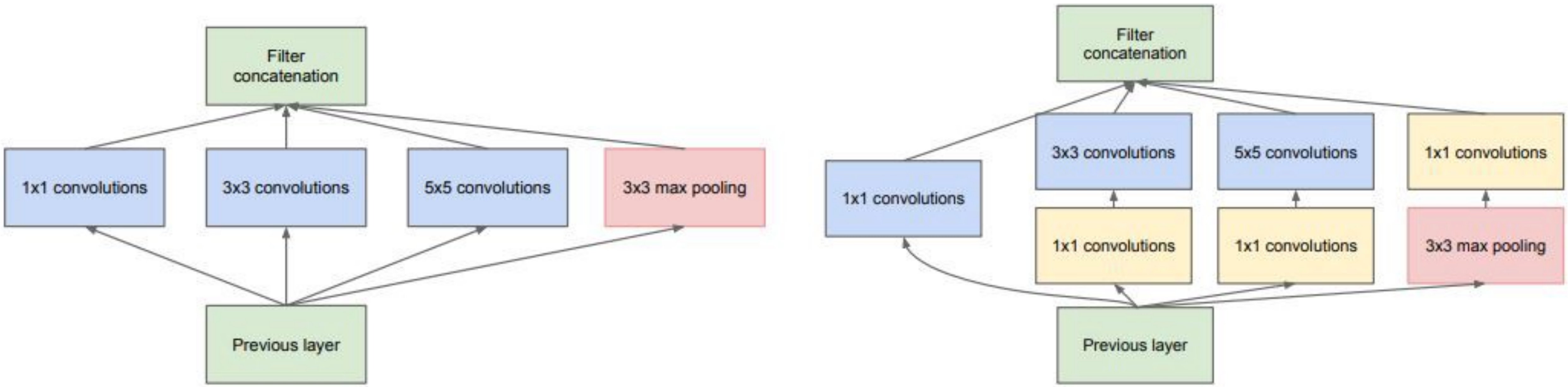


- The idea came out as a solution to an observation — Deep neural networks perform worse as we keep on adding layer.

- Instead of learning mapping between output of layer and its input, learn the difference between them — learn the residual.

- The vanishing gradients problem which usually make deep neural networks numb to learning were removed.

GoogLeNet / Inception



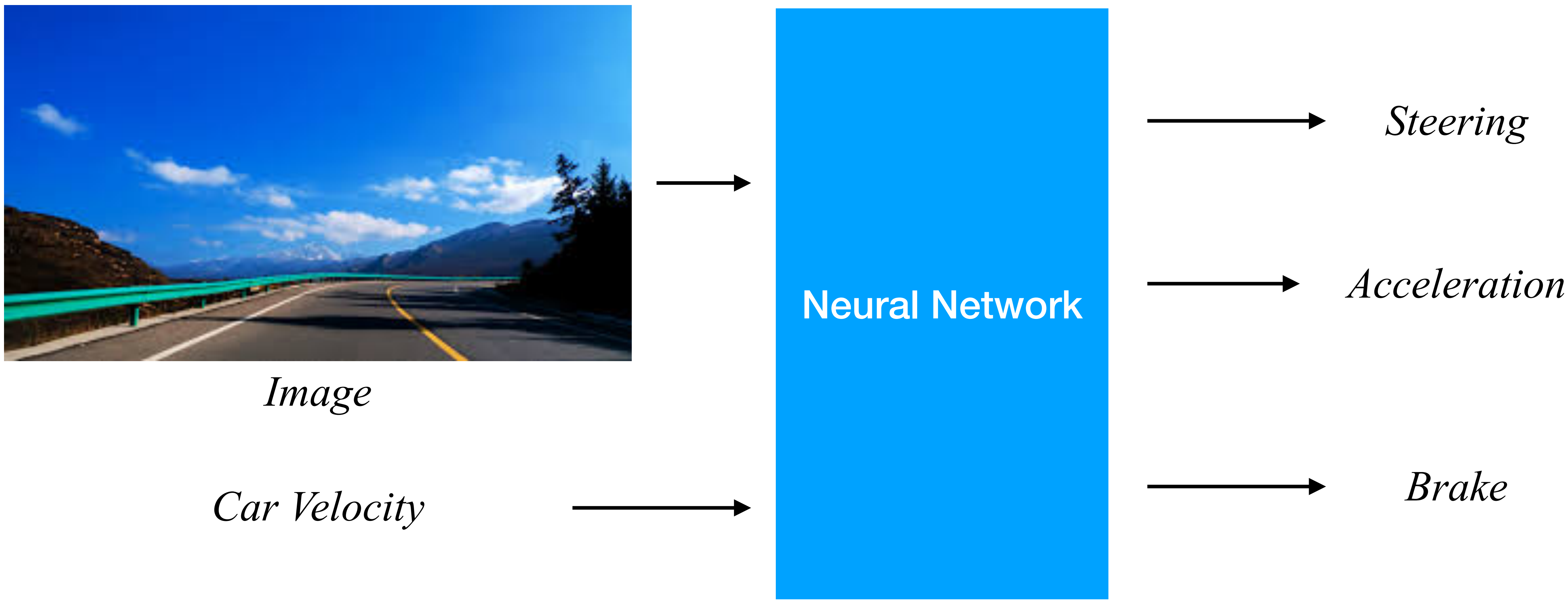
(a) Inception module, naïve version (b) Inception module with dimension reductions

Reasons for using these inception modules:

- Each layer type extracts different information from input. Information gathered from a 3×3 layer will differ from information gathered from a 5×5 layer. How do we know which transformation will be the best at a given layer? So we use them all.

- **Dimensionality reduction using 1×1 convolutions.** Consider a 128x128x256 input. If we pass it through 20 filters of size 1×1, we will get an output of 128x128x20. So we apply them before the 3×3 or 5×5 convolutions to decrease the number of input filters to these layers in the inception block used for dimensionality reduction.

Custom Neural Networks



- We need multiple inputs and outputs in this case.

Multiple Inputs and Outputs in Keras

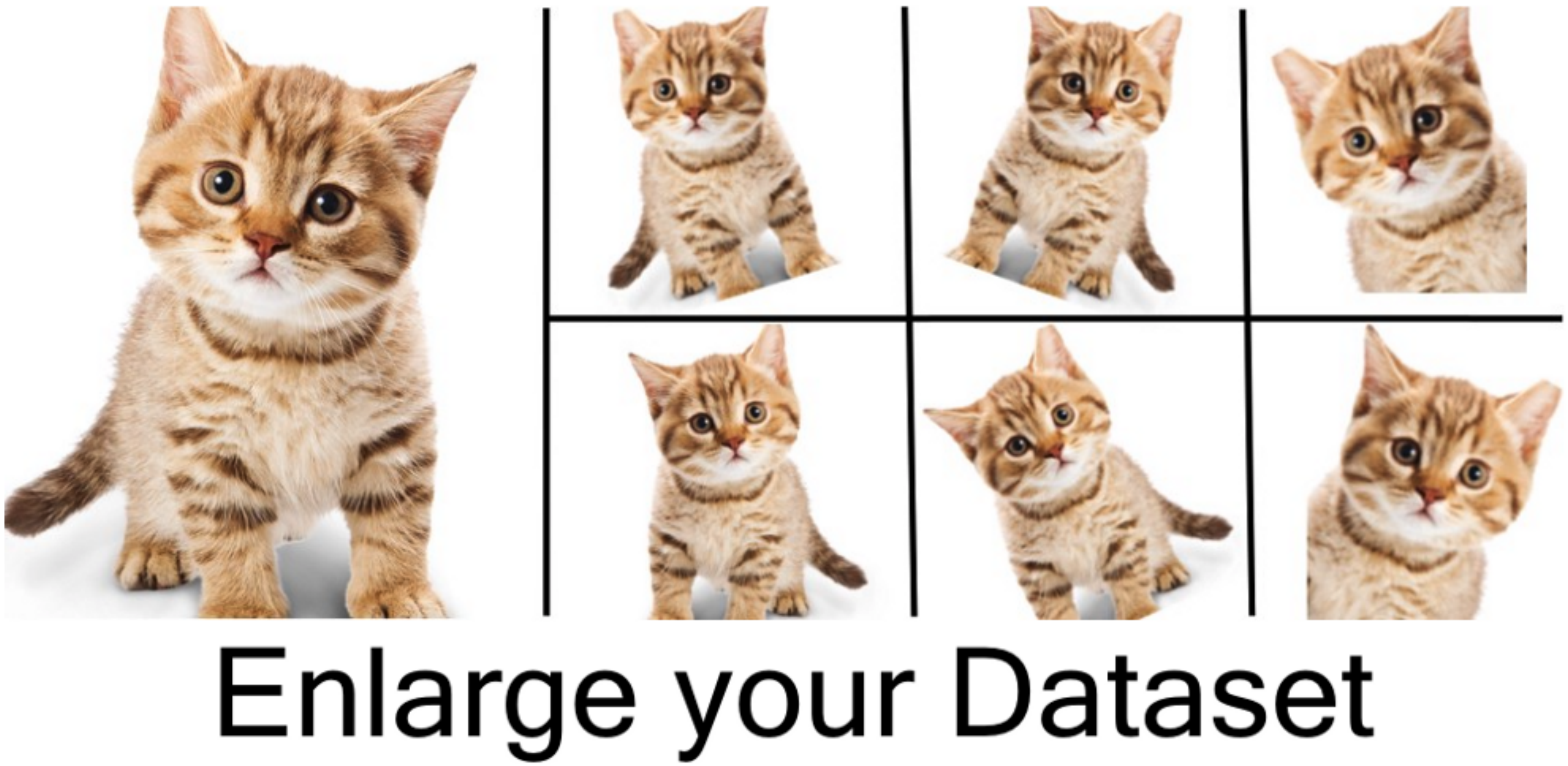

```
from keras.models import Sequential, Model, load_model
from keras.layers import Conv2D, Dense, Input, Concatenate, MaxPooling2D, Flatten
from keras.utils import to_categorical
from keras.callbacks import EarlyStopping, ModelCheckpoint, CSVLogger, ReduceLROnPlateau, Callback, TensorBoard
from keras.regularizers import l2

image_input = Input(shape=(images.shape[1],images.shape[2],images.shape[3],))
age_input = Input(shape=(1,))

c = Conv2D(32,(3,3),padding='same',activation='relu')(image_input)
c = MaxPooling2D(pool_size=(2,2))(c)

c = Conv2D(16,(3,3),padding='same',activation='relu')(c)
c = MaxPooling2D(pool_size=(2,2))(c)

c = Flatten()(c)

concat = Concatenate()([c, age_input])
x = Dense(64, activation='relu')(concat)
x = Dense(32, activation='relu')(x)

y = Dense(32, activation='relu')(concat)
y = Dense(16, activation='relu')(y)

emotion_pred = Dense(len(EMOTIONS), activation='softmax')(x)
gender_pred = Dense(2, activation='softmax')(y)

model = Model(inputs=[image_input, age_input], outputs=[emotion_pred, gender_pred])
model.summary()
```

– Suppose we have images of faces and age information and we want to estimate the emotion and the gender from the given input image and the age.

Using Transfer Learning on certain Branch

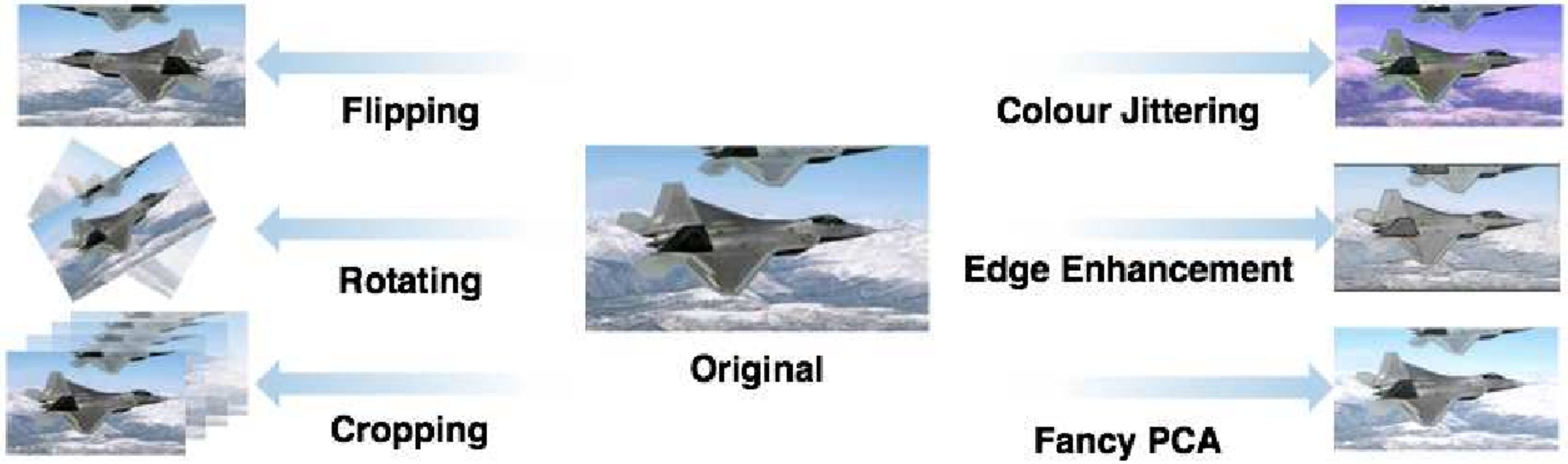
```python
from keras.models import Sequential, Model, load_model
from keras.layers import Conv2D, Dense, Input, Concatenate, MaxPooling2D, Flatten
from keras.utils import to_categorical
from keras.callbacks import EarlyStopping, ModelCheckpoint, CSVLogger, ReduceLROnPlateau, Callback, TensorBoard
from keras.regularizers import l2
from keras.applications.resnet50 import ResNet50

# image_input = Input(shape=(images.shape[1],images.shape[2],images.shape[3],))
age_input = Input(shape=(1,))

# c = Conv2D(32,(3,3),padding='same',activation='relu')(image_input)
# c = MaxPooling2D(pool_size=(2,2))(c)

# c = Conv2D(16,(3,3),padding='same',activation='relu')(c)
# c = MaxPooling2D(pool_size=(2,2))(c)

# c = Flatten()(c)

base_model = ResNet50(include_top=False, weights='imagenet', input_tensor=None,
                      input_shape=(images.shape[1],images.shape[2],images.shape[3],), pooling='avg')

concat = Concatenate()([base_model.output, age_input])
x = Dense(64, activation='relu')(concat)
x = Dense(32, activation='relu')(x)

y = Dense(32, activation='relu')(concat)
y = Dense(16, activation='relu')(y)

emotion_pred = Dense(len(EMOTIONS), activation='softmax')(x)
gender_pred = Dense(2, activation='softmax')(y)

model = Model(inputs=[base_model.input, age_input], outputs=[emotion_pred, gender_pred])
model.summary()
```

- All you need to do is design the inputs and the outputs.

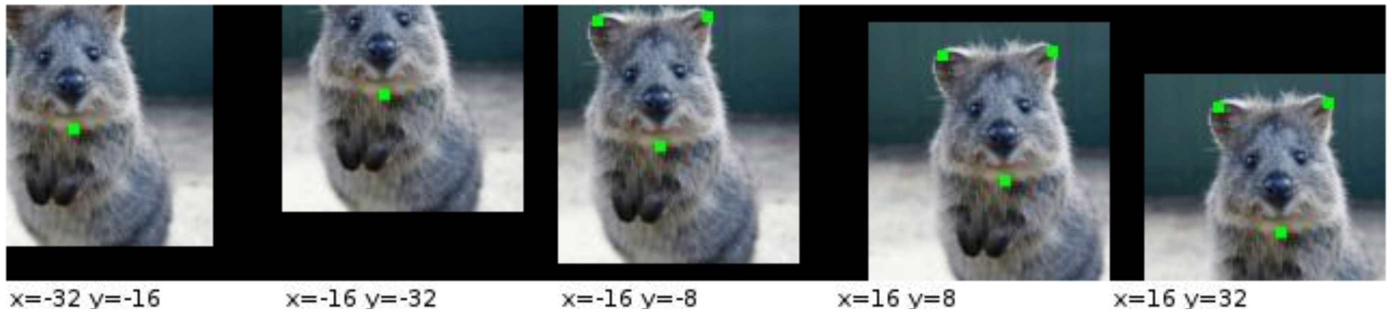


- One way of reducing overfitting.

Data Augmentation



Affine: Translate



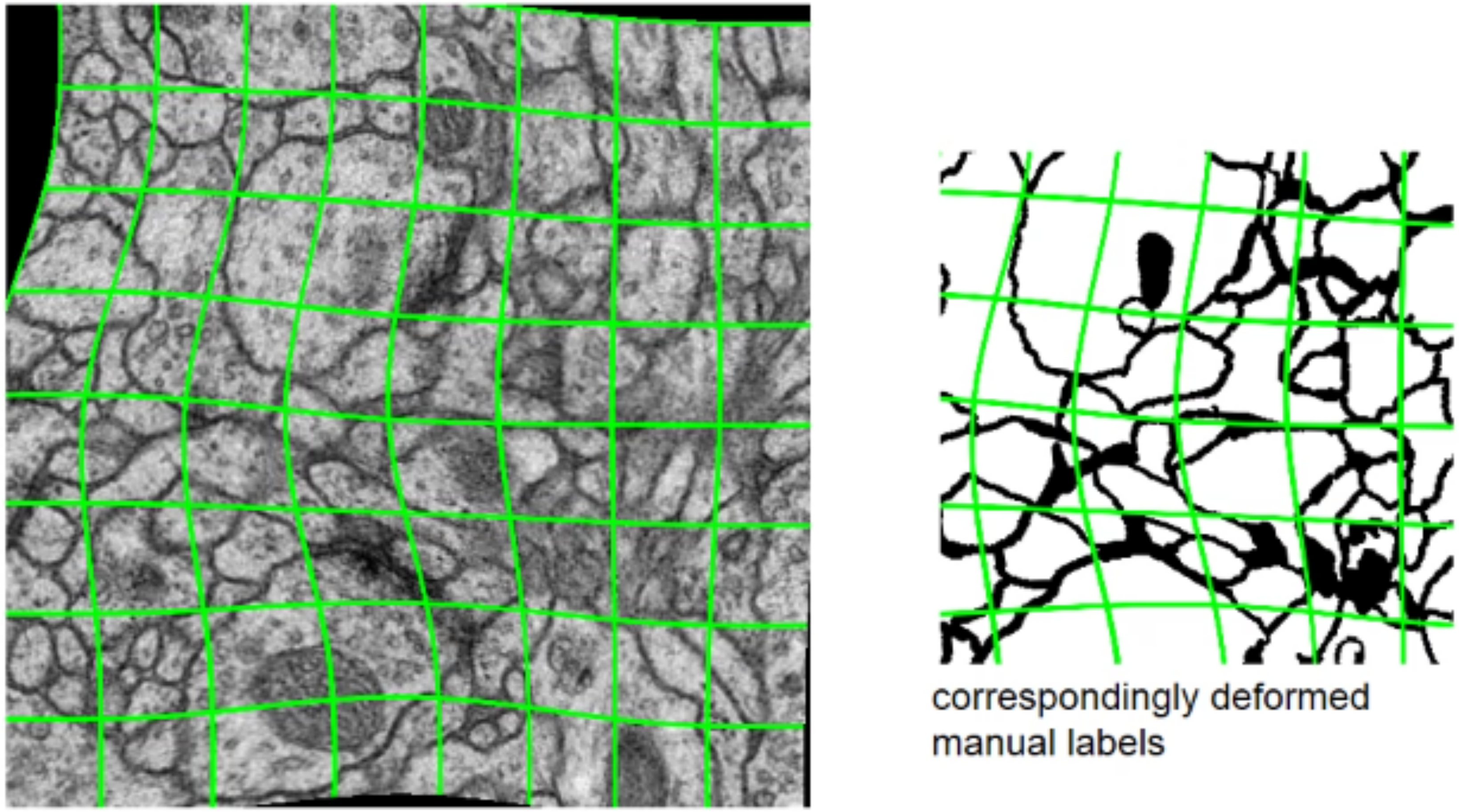
Affine: Shear

Data Augmentation

5 Minute Teaser Presentation of the U-net

Augment Training Data using Deformations

UNI FREIBURG

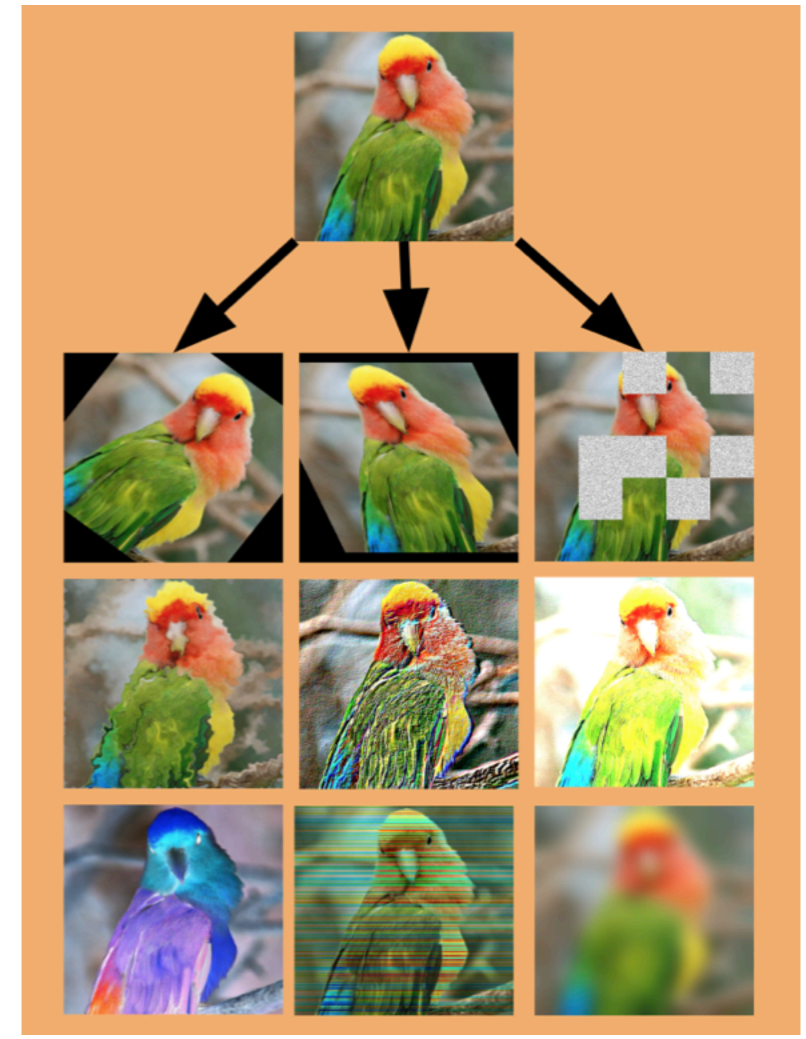


correspondingly deformed manual labels

resulting deformed image
(for visualization: no rotation, no shift, no extrapolation)

Olaf Ronneberger, University of Freiburg, Germany, 22.5.2015 14

Download video: u-net-teaser.mp4 (68MB)



Data Augmentation using Keras

```python
from keras.preprocessing.image import ImageDataGenerator
from keras.utils import np_utils
from keras.callbacks import ReduceLROnPlateau, CSVLogger, EarlyStopping, ModelCheckpoint
```

```python
lr_reducer = ReduceLROnPlateau(factor=np.sqrt(0.1), cooldown=0, patience=5, min_lr=0.5e-6)
early_stopper = EarlyStopping(min_delta=0.001, patience=20)
csv_logger = CSVLogger('./model/'+tag+'_log.csv')
model_checkpoint = ModelCheckpoint('./model/'+tag+'-{epoch:02d}-{val_acc:.2f}.hdf5', monitor='val_loss', verbose=1,
                                   save_best_only=True, save_weights_only=False, mode='min', period=1)
```

```python
        # This will do preprocessing and realtime data augmentation:
        datagen = ImageDataGenerator(
            featurewise_center=False,  # set input mean to 0 over the dataset
            samplewise_center=False,  # set each sample mean to 0
            featurewise_std_normalization=False,  # divide inputs by std of the dataset
            samplewise_std_normalization=False,  # divide each input by its std
            zca_whitening=False,  # apply ZCA whitening
            rotation_range=10,  # randomly rotate images in the range (degrees, 0 to 180)
            width_shift_range=0.1,  # randomly shift images horizontally (fraction of total width)
            height_shift_range=0.1,  # randomly shift images vertically (fraction of total height)
            horizontal_flip=True,  # randomly flip images
            # brightness_range=[0.8, 1.2],
            vertical_flip=False)  # randomly flip images

        # Compute quantities required for featurewise normalization
        # (std, mean, and principal components if ZCA whitening is applied).
        datagen.fit(X_train)

        model.fit_generator(datagen.flow(X_train, y_train, batch_size=batch_size),
                            steps_per_epoch=X_train.shape[0] // batch_size,
                            validation_data=(X_val, y_val),
                            epochs=nb_epoch, verbose=1, max_q_size=100,
                            callbacks=[lr_reducer, csv_logger, model_checkpoint, early_stopper])
```

Assignment – Data Augmentation in custom generator

```python
# Example of data generator for the assignment
def data_generator(images, amt=10):
    _images = images[amt]
    # Do something

    yield _images

data_gen = data_generator(images)
img = next(data_gen)
img = next(data_gen)
print(img)
```

- Submit your .py code. No report this time.
- You can use any kind of images, but I recommend you to use car dataset for your class project.
- Use three different image processing techniques to do the data augmentation using the code above.