Artificial Neural Networks and Deep Learning

Keras tutorial - 04/11/2020

Francesco Lattari, PhD student (francesco.lattari@polimi.it)

Artificial Intelligence and Robotics Laboratory

Dealing with Overfitting

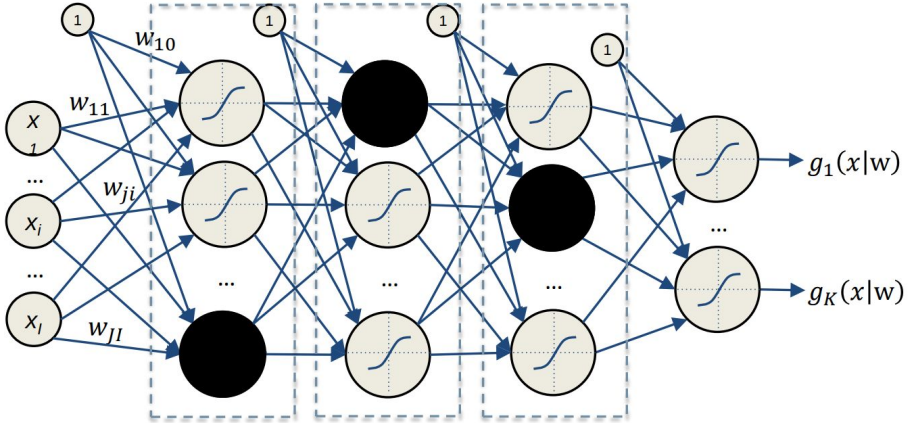
- Reduce number of parameters
- Dropout
- Weight decay
- Early stopping

- Reduce number of parameters
- Dropout
- Weight decay
- Early stopping

Class tf.keras.layers.Dropout

https://www.tensorflow.org/api_docs/python/tf/keras/layers/Dropout

Each hidden unit is turned off with probability equal to ‘rate’ (main parameter)



- Reduce number of parameters
- Dropout
- Weight decay
- Early stopping

Class tf.keras.regularizers.l2

https://www.tensorflow.org/api_docs/python/tf/keras/regularizers/l2

$$argmin_w \underbrace{\sum_{n=1}^{N} \left(t_n - g(x_n|w)\right)^2}_{\text{Fitting}} + \underbrace{\gamma \sum_{q=1}^{Q} \left(w_q\right)^2}_{\text{Regularization}}$$

- e.g., fully-connected layer

```
tf.keras.layers.Dense(
    units,
    activation=None,
    use_bias=True,
    kernel_initializer='glorot_uniform',
    bias_initializer='zeros',
    kernel_regularizer=tf.keras.regularizers.l2(0.001),
    bias_regularizer=None,
    activity_regularizer=None,
    kernel_constraint=None,
    bias_constraint=None,
    **kwargs,
)
```

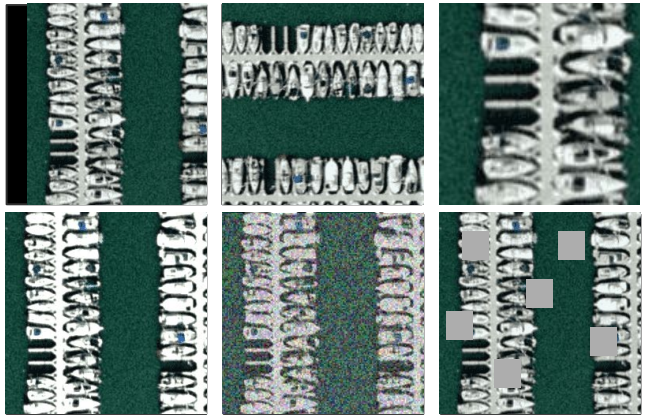
- Reduce number of parameters
- Dropout
- Weight decay
- Early stopping

Class tf.keras.callbacks.EarlyStopping

https://www.tensorflow.org/api_docs/python/tf/keras/callbacks/EarlyStopping

```
EarlyStopping(
  monitor='val_loss',
  min_delta=0,
  patience=0,
  verbose=0,
  mode='auto',
  baseline=None,
  restore_best_weights=False
)
```

- Reduce number of parameters
- Dropout
- Weight decay
- Early stopping
- Data augmentation



Class tf.keras.preprocessing.image.ImageDataGenerator

https://www.tensorflow.org/api_docs/python/tf/keras/preprocessing/image/ImageDataGenerator

```
ImageDataGenerator(
    featurewise_center=False,
    samplewise_center=False,
    featurewise_std_normalization=False,
    samplewise_std_normalization=False,
    zca_whitening=False,
    zca_epsilon=1e-06,
    rotation_range=0,
    width_shift_range=0.0,
    height_shift_range=0.0,
    brightness_range=None,
    shear_range=0.0,
    zoom_range=0.0,
    channel_shift_range=0.0,
    fill_mode='nearest',
    cval=0.0,
    horizontal_flip=False,
    vertical_flip=False,
    rescale=None,
    preprocessing_function=None,
    data_format=None,
    validation_split=0.0,
    dtype=None)
```

Data Loader from directory

1. Organize dataset folders:

- data/
  - training/
    - class_1/
      - img1, img2, …, imgN
    - …
    - class_K/
      - img1, img2, ... , imgN
  - validation/
    - class_1/
      - img1, img2, …, imgN
    - …
    - class_K/
      - img1, img2, …, imgN
  - test/
    - class_1/
      - img1, img2, …, imgN
    - ...
    - class_K/
      - img1, img2, …, imgN

Data Loader from directory

2. Initialize ImageDataGenerator:

```
img_gen = ImageDataGenerator(...)
```

Data Loader from directory

2. Initialize ImageDataGenerator:

```
img_gen = ImageDataGenerator(...)
```

3. Call flow_from_directory class method:

```
flow_img_gen = flow_from_directory(
      directory,
      target_size=(256, 256),
      color_mode='rgb',
      classes=None,
      class_mode='categorical',
      batch_size=32,
      shuffle=True,
      seed=None,
      save_to_dir=None,
      save_prefix='',
      save_format='png',
      follow_links=False,
      subset=None,
      interpolation='nearest')
```

Data Loader from directory

2. Initialize ImageDataGenerator:

```
img_gen = ImageDataGenerator(...)
```

3. Call flow_from_directory class method:

```
flow_img_gen = flow_from_directory(
      directory,
      target_size=(256, 256),
      color_mode='rgb',
      classes=None,
      class_mode='categorical',
      batch_size=32,
      shuffle=True,
      seed=None,
      save_to_dir=None,
      save_prefix='',
      save_format='png',
      follow_links=False,
      subset=None,
      interpolation='nearest')
```

4. a. Create a tf.data.Dataset object

```
dataset = tf.data.Dataset.from_generator(
      flow_img_gen)
```

b. Use generator directly

```
tf.keras.Model.fit_generator(
      generator,
      steps_per_epoch=None,
      epochs=1,
      verbose=1,
      callbacks=None,
      validation_data=None,
      validation_steps=None,
      validation_freq=1,
      class_weight=None,
      max_queue_size=10,
      workers=1,
      use_multiprocessing=False,
      shuffle=True,
      initial_epoch=0)
```

UC Merced Land Use Dataset

http://weegee.vision.ucmerced.edu/datasets/landuse.html

21 classes, 100 images each:

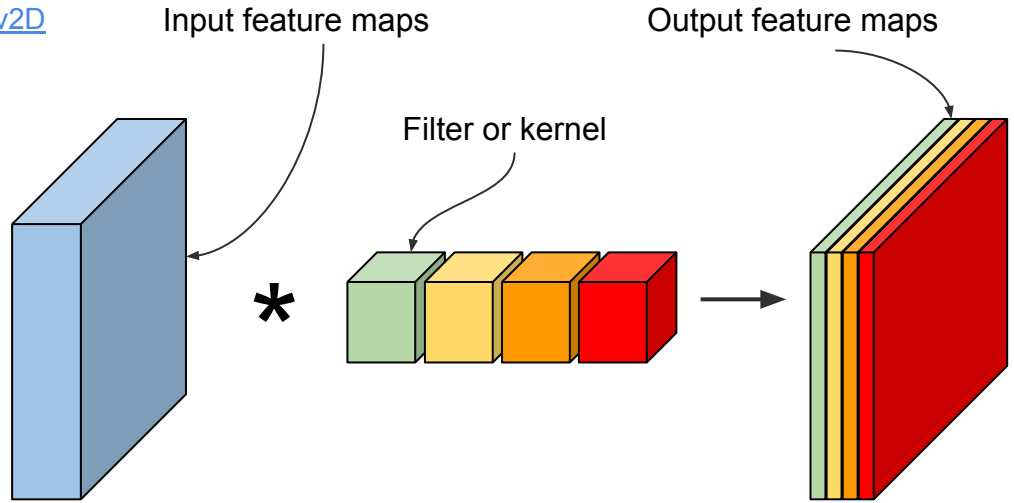
- agricultural
- airplane
- baseballdiamond
- beach
- buildings
- chaparral
- denseresidential
- forest
- freeway
- golfcourse
- harbor
- intersection
- mediumresidential
- mobilehomepark
- overpass
- parkinglot
- river
- runway
- sparseresidential
- storagetanks
- tenniscourt

Image size: 256x256 pixels

tf.keras.layers.Conv2D

https://www.tensorflow.org/api_docs/python/tf/keras/layers/Conv2D

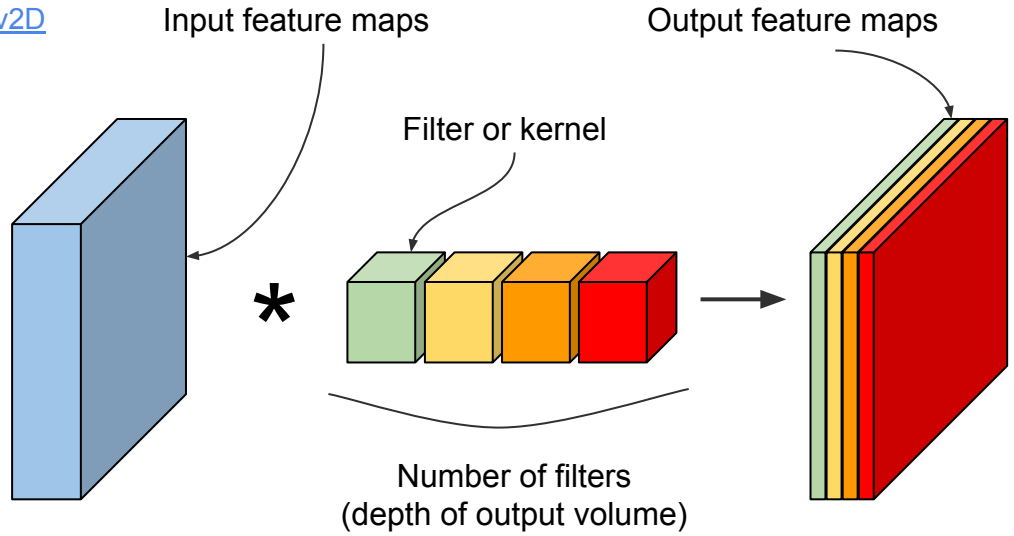
```
Conv2D(
    filters,
    kernel_size,
    strides=(1, 1),
    padding='valid',
    data_format=None,
    dilation_rate=(1, 1),
    activation=None,
    use_bias=True,
    kernel_initializer='glorot_uniform',
    bias_initializer='zeros',
    kernel_regularizer=None,
    bias_regularizer=None,
    activity_regularizer=None,
    kernel_constraint=None,
    bias_constraint=None,
    **kwargs)
```



tf.keras.layers.Conv2D

https://www.tensorflow.org/api_docs/python/tf/keras/layers/Conv2D

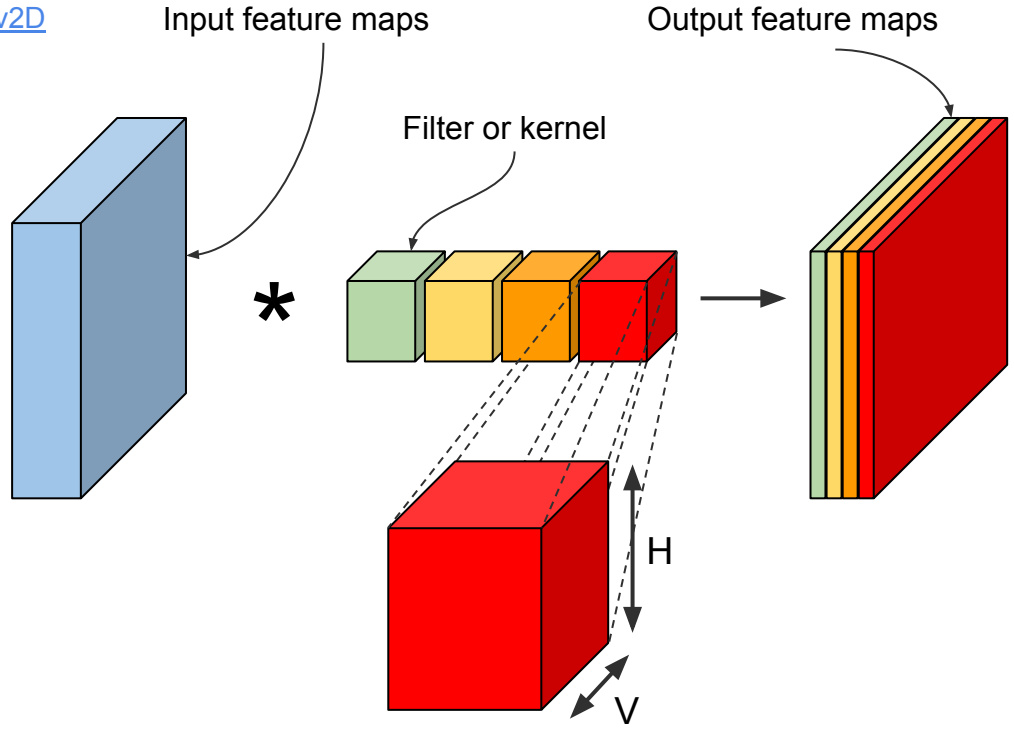
```
Conv2D(
    filters,
    kernel_size,
    strides=(1, 1),
    padding='valid',
    data_format=None,
    dilation_rate=(1, 1),
    activation=None,
    use_bias=True,
    kernel_initializer='glorot_uniform',
    bias_initializer='zeros',
    kernel_regularizer=None,
    bias_regularizer=None,
    activity_regularizer=None,
    kernel_constraint=None,
    bias_constraint=None,
    **kwargs)
```



tf.keras.layers.Conv2D

https://www.tensorflow.org/api_docs/python/tf/keras/layers/Conv2D

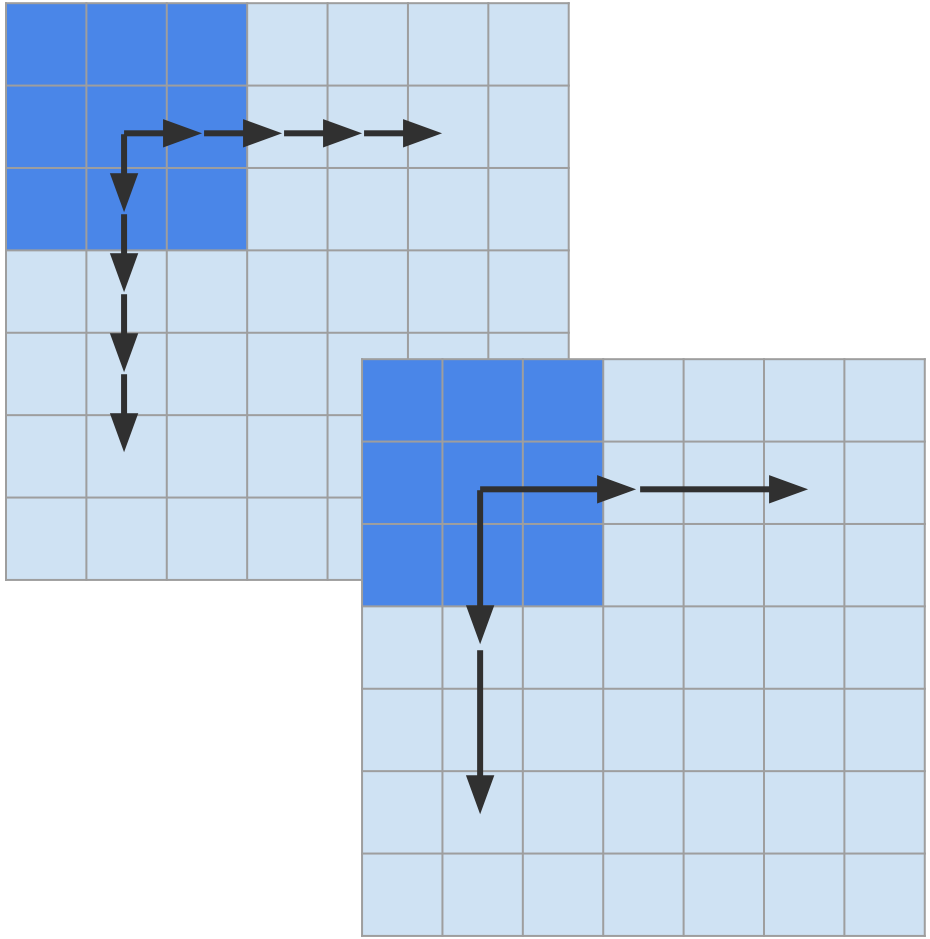
```
Conv2D(
    filters,
    kernel_size,
    strides=(1, 1),
    padding='valid',
    data_format=None,
    dilation_rate=(1, 1),
    activation=None,
    use_bias=True,
    kernel_initializer='glorot_uniform',
    bias_initializer='zeros',
    kernel_regularizer=None,
    bias_regularizer=None,
    activity_regularizer=None,
    kernel_constraint=None,
    bias_constraint=None,
    **kwargs)
```



tf.keras.layers.Conv2D

https://www.tensorflow.org/api_docs/python/tf/keras/layers/Conv2D

```
Conv2D(
    filters,
    kernel_size,
    strides=(1, 1),
    padding='valid',
    data_format=None,
    dilation_rate=(1, 1),
    activation=None,
    use_bias=True,
    kernel_initializer='glorot_uniform',
    bias_initializer='zeros',
    kernel_regularizer=None,
    bias_regularizer=None,
    activity_regularizer=None,
    kernel_constraint=None,
    bias_constraint=None,
    **kwargs)
```

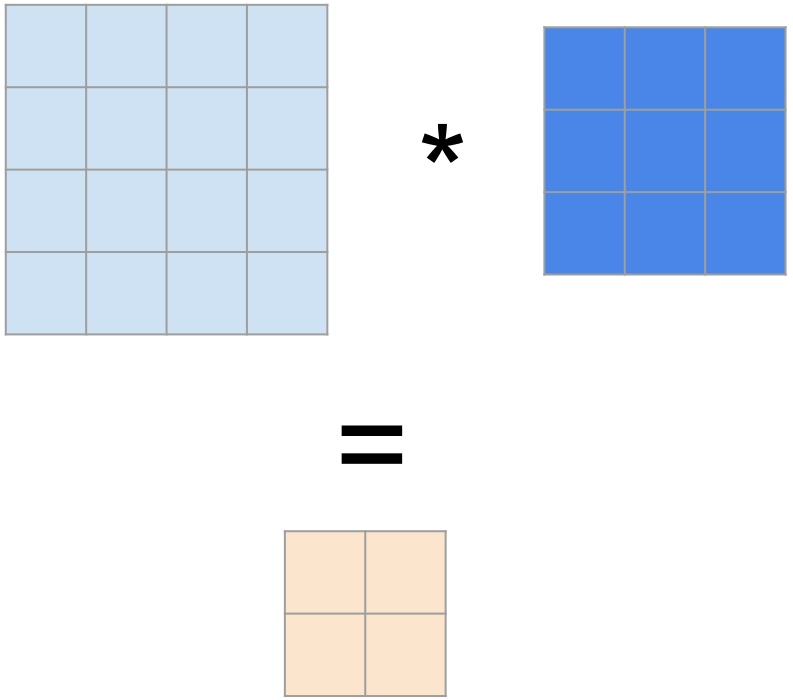


‘valid’
(stride 1)

tf.keras.layers.Conv2D

https://www.tensorflow.org/api_docs/python/tf/keras/layers/Conv2D

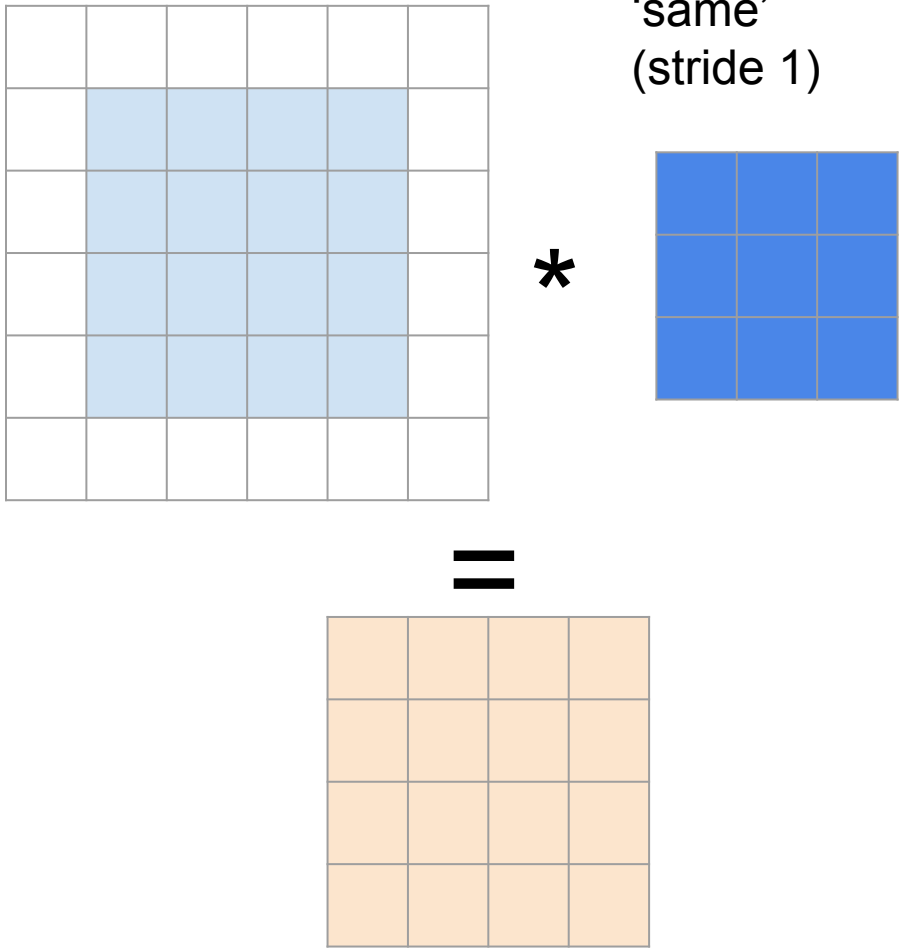
```
Conv2D(
    filters,
    kernel_size,
    strides=(1, 1),
    padding='valid',
    data_format=None,
    dilation_rate=(1, 1),
    activation=None,
    use_bias=True,
    kernel_initializer='glorot_uniform',
    bias_initializer='zeros',
    kernel_regularizer=None,
    bias_regularizer=None,
    activity_regularizer=None,
    kernel_constraint=None,
    bias_constraint=None,
    **kwargs)
```



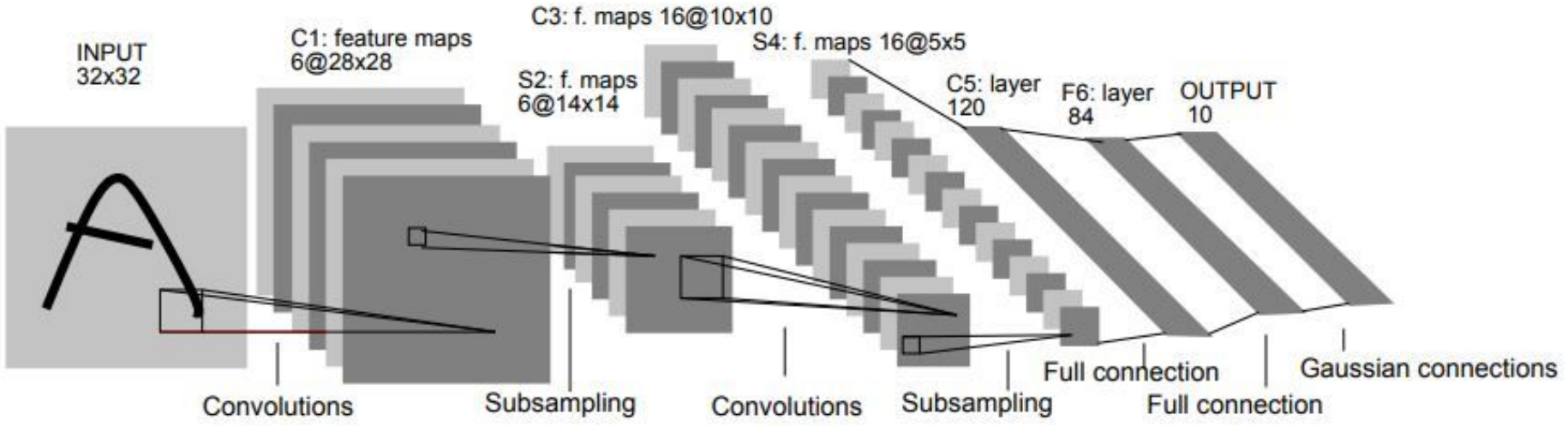
tf.keras.layers.Conv2D

https://www.tensorflow.org/api_docs/python/tf/keras/layers/Conv2D

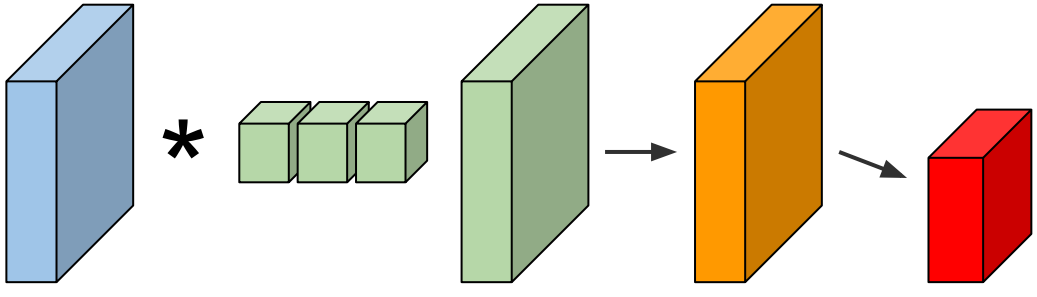
```
Conv2D(
    filters,
    kernel_size,
    strides=(1, 1),
    padding='same',
    data_format=None,
    dilation_rate=(1, 1),
    activation=None,
    use_bias=True,
    kernel_initializer='glorot_uniform',
    bias_initializer='zeros',
    kernel_regularizer=None,
    bias_regularizer=None,
    activity_regularizer=None,
    kernel_constraint=None,
    bias_constraint=None,
    **kwargs)
```



Convolutional Neural Network
Typical architecture



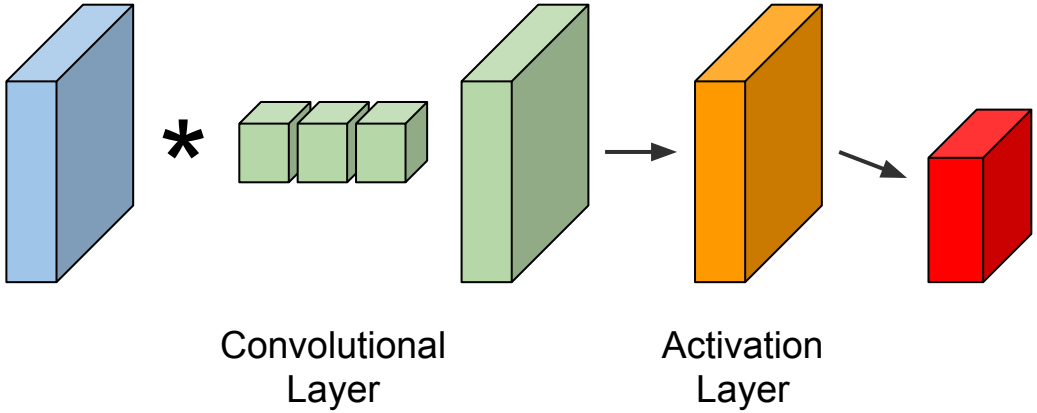
LeNet-5 (1998) https://ieeexplore.ieee.org/document/726791



Convolutional Layer

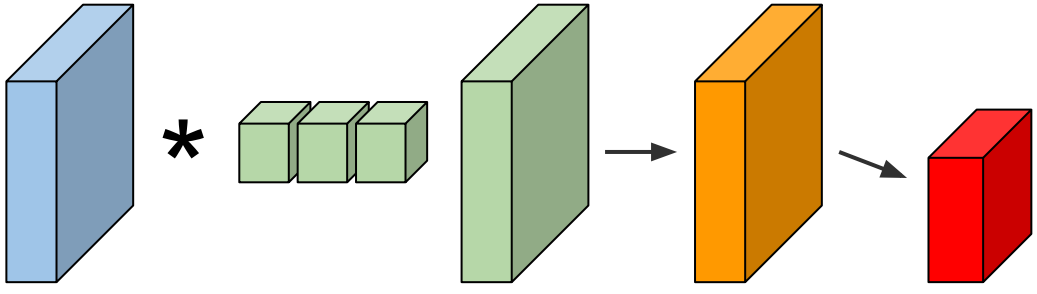
tf.keras.layers.Conv2D

https://www.tensorflow.org/api_docs/python/tf/keras/layers/Conv2D



tf.keras.activations

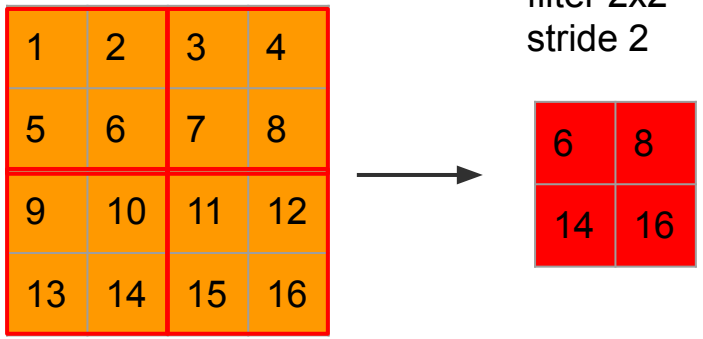
https://www.tensorflow.org/api_docs/python/tf/keras/activations



Convolutional Layer | Activation Layer | Pooling Layer

tf.keras.layers.MaxPool2D (typically used)

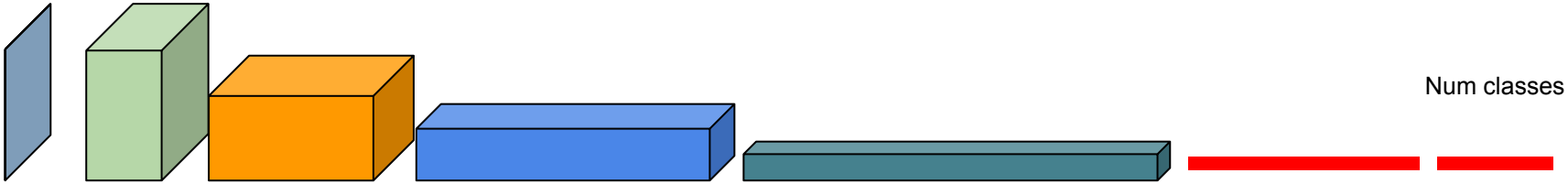
https://www.tensorflow.org/api_docs/python/tf/keras/layers/MaxPool2D



Max Pooling
filter 2x2
stride 2

As the network goes deeper:

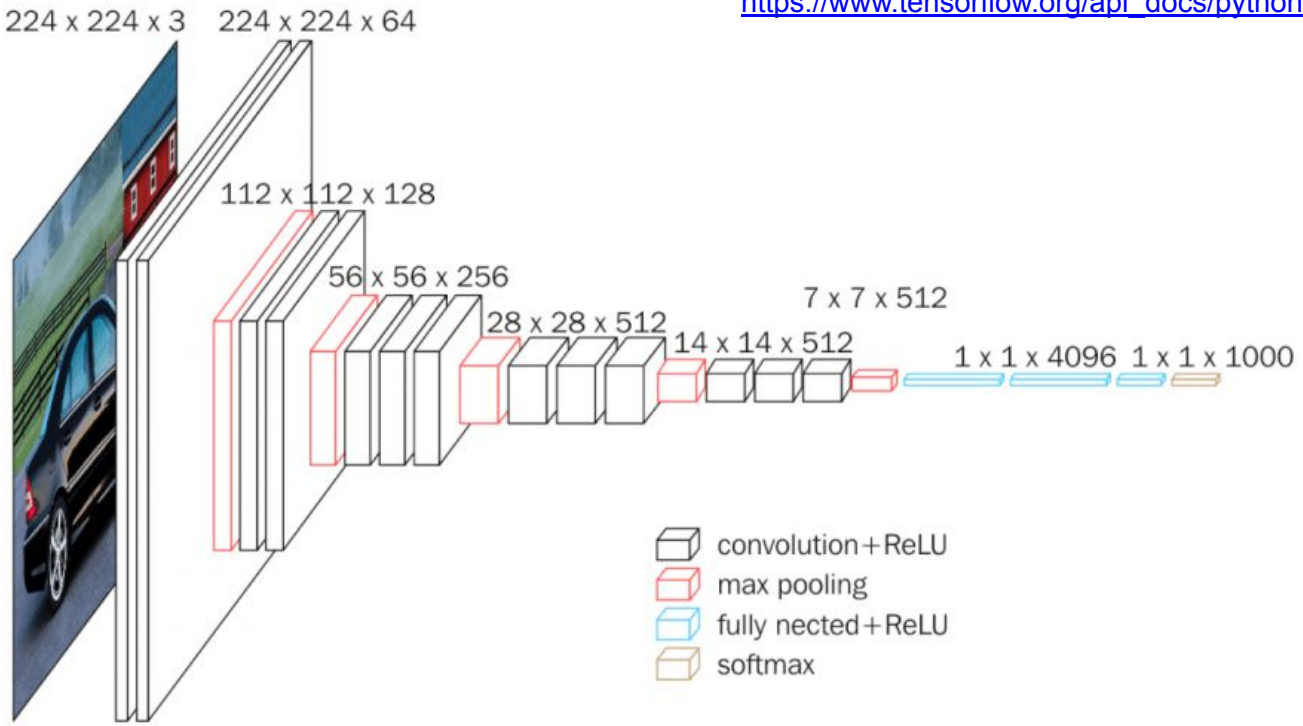
- Reduce spatial dimension (downsampling)
- Increase number of filters (low-level features -> mid-level features -> high-level features)



- Top layers: Fully-Connected Layers

Other models in
tf.keras.applications

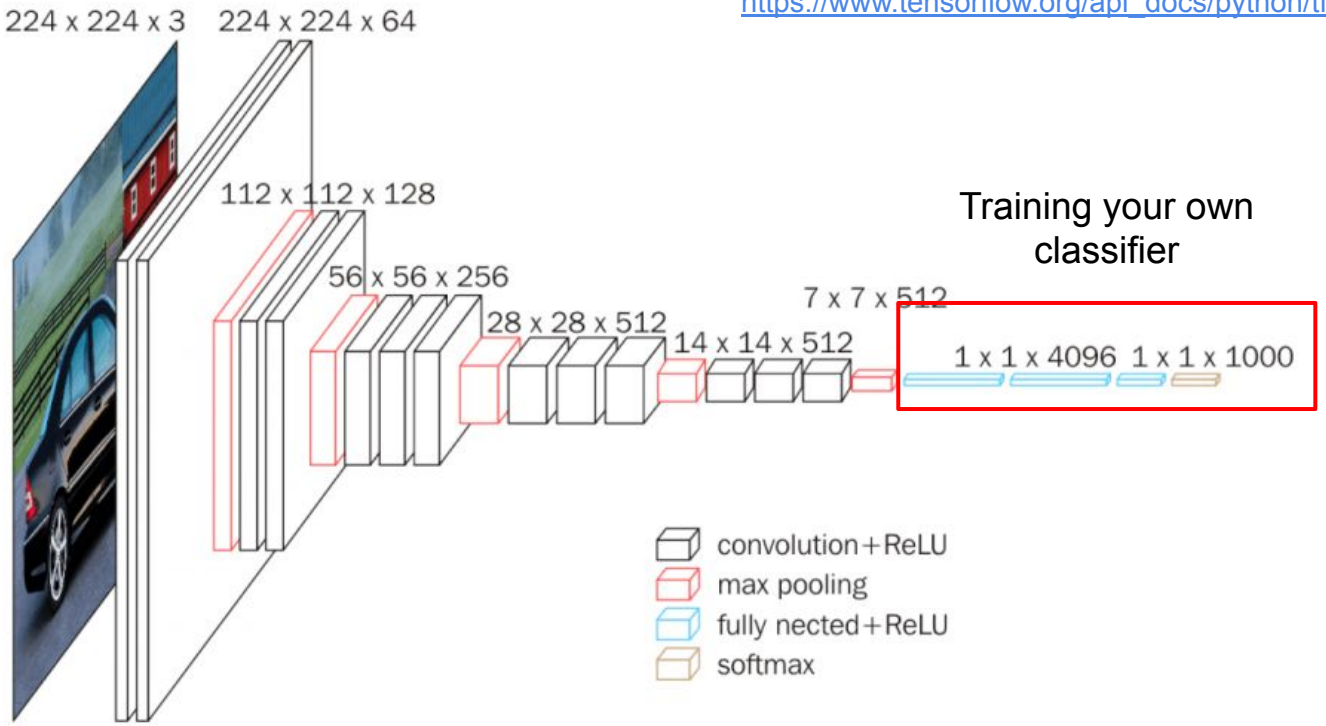
https://www.tensorflow.org/api_docs/python/tf/keras/applications



VGG16 https://arxiv.org/abs/1409.1556

Other models in
tf.keras.applications

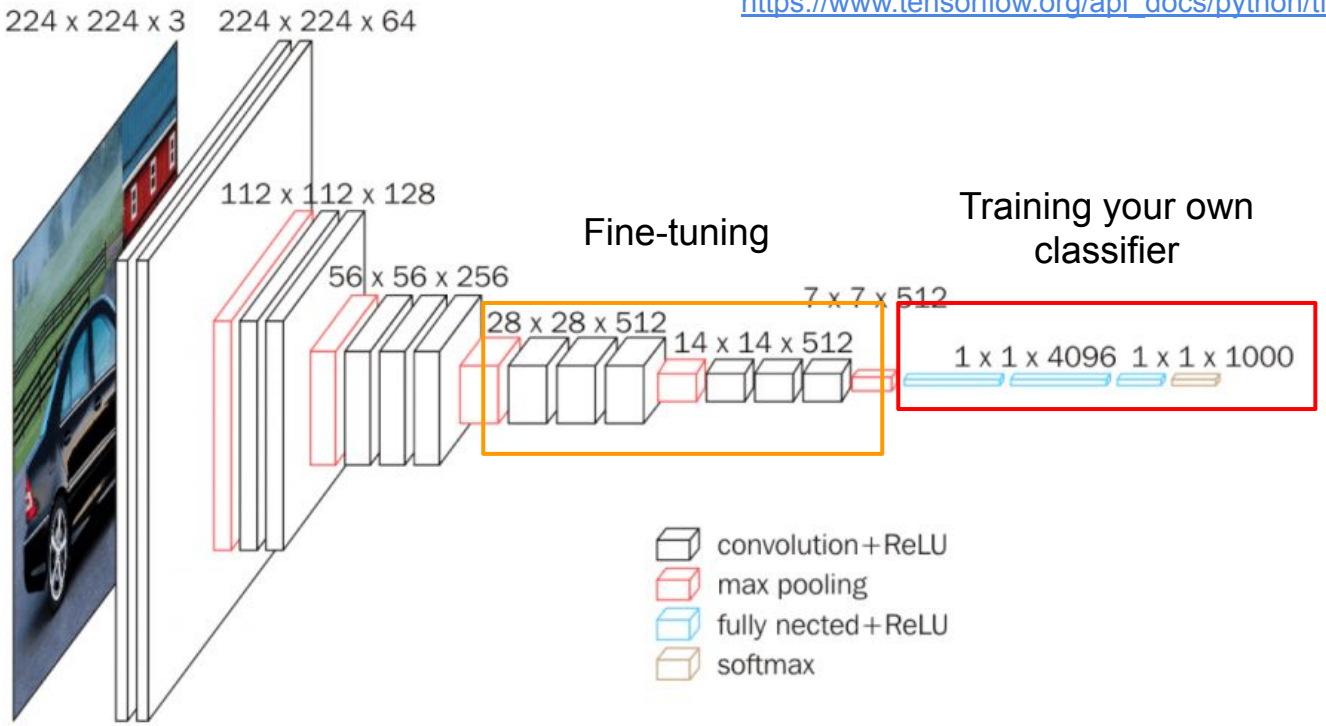
https://www.tensorflow.org/api_docs/python/tf/keras/applications



VGG16 https://arxiv.org/abs/1409.1556

Other models in
tf.keras.applications

https://www.tensorflow.org/api_docs/python/tf/keras/applications



VGG16 https://arxiv.org/abs/1409.1556