ECE449 Lab 3

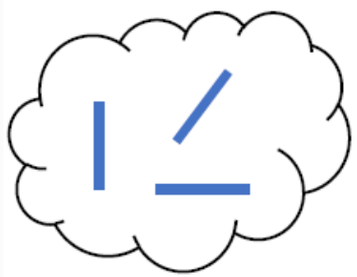
October 4th, 2023

Lab 2

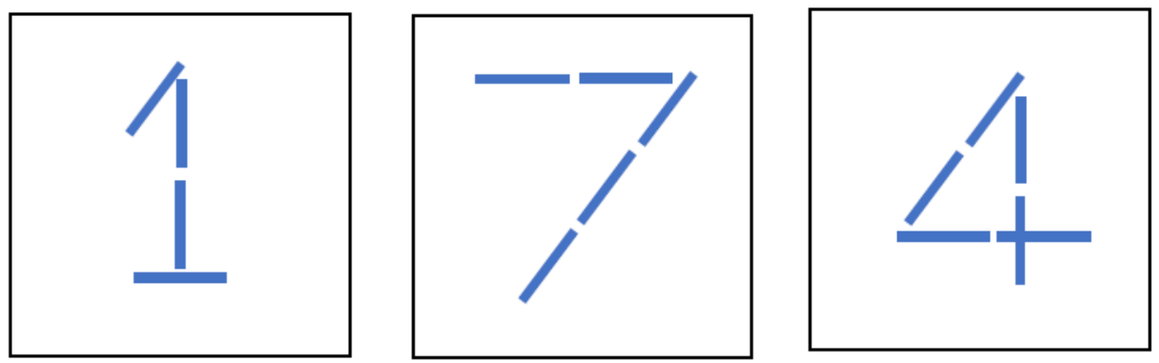
Convolutional Neural Networks (CNN)

- Build invariance to certain variations shifting, illumination, …
- Deal with very high dimensional inputs without the need of large number of parameters
- Instead of learning to detect the whole image, detect smaller fragments in the image.

Lab 2

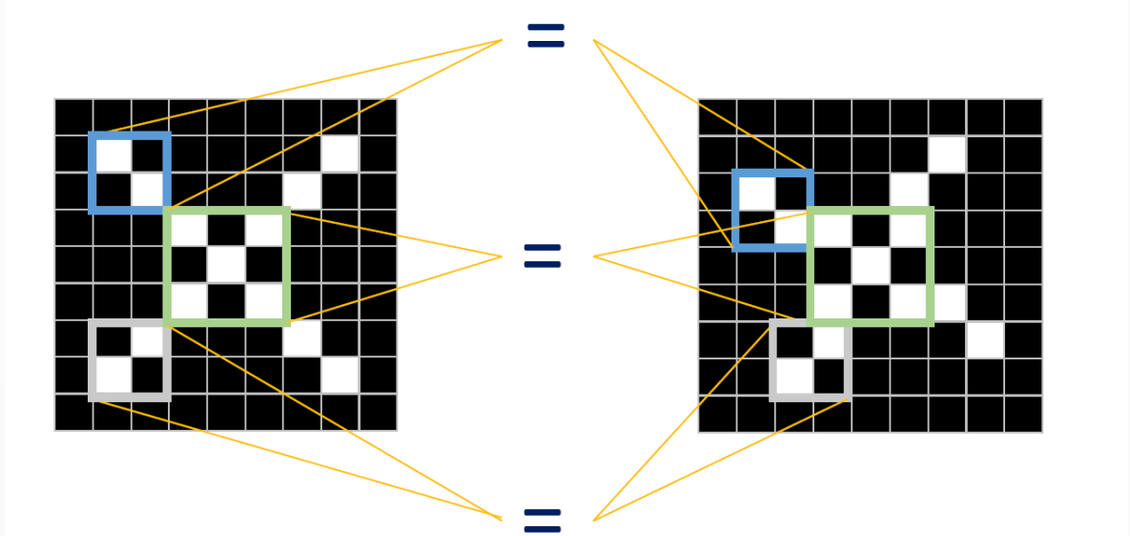


A set of curves



Images are made of three basic curves

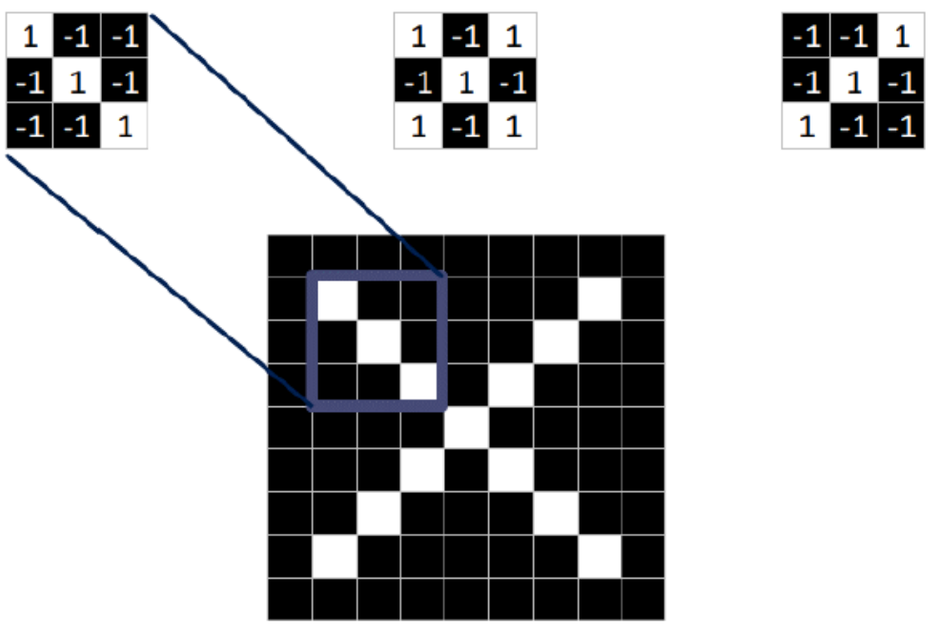
- Understanding that these two images are two different instances (i.e. ‘X’) of the same concept is difficult for a simple neural network.



Images from Samrudhdhi Rangrej’s presentation

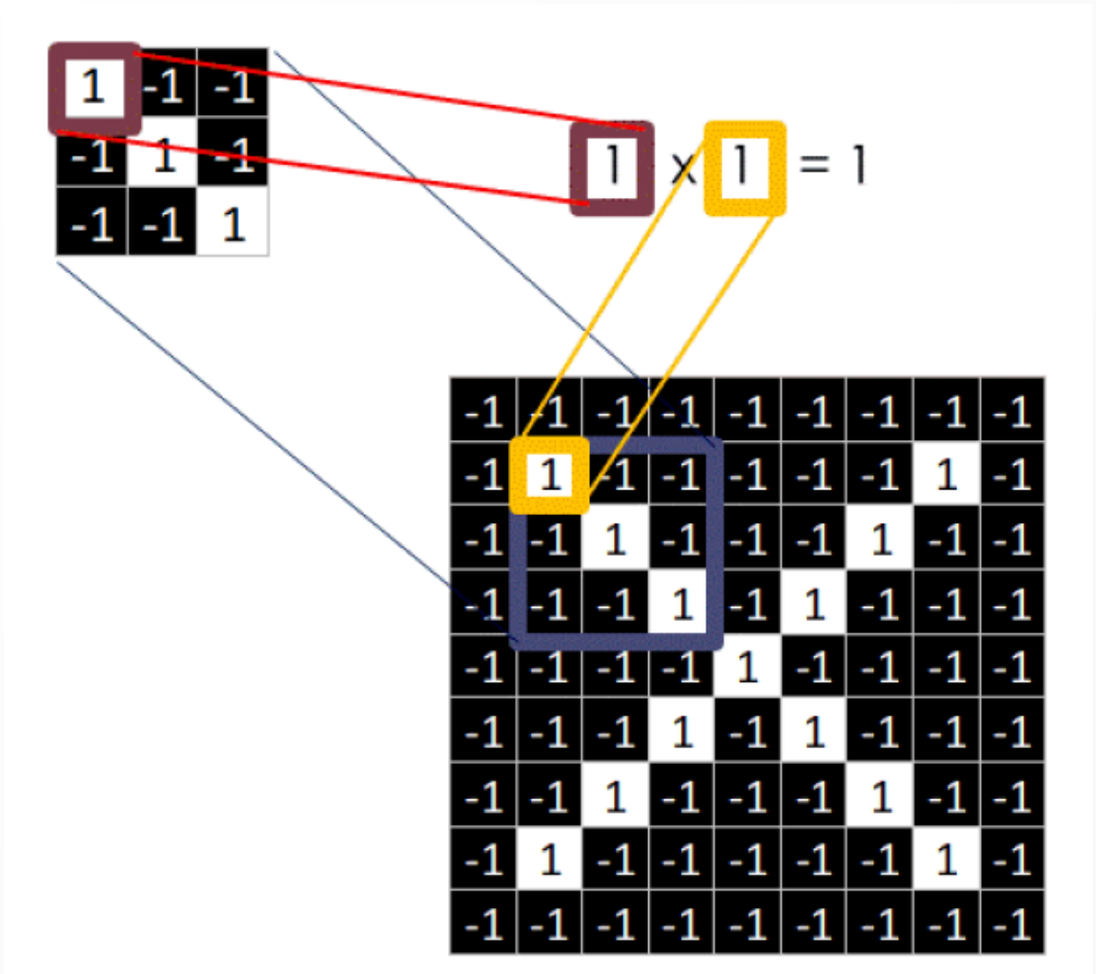
Lab 2

- A parameter matrix is called a filter.
- We need three filters to detect three curves.



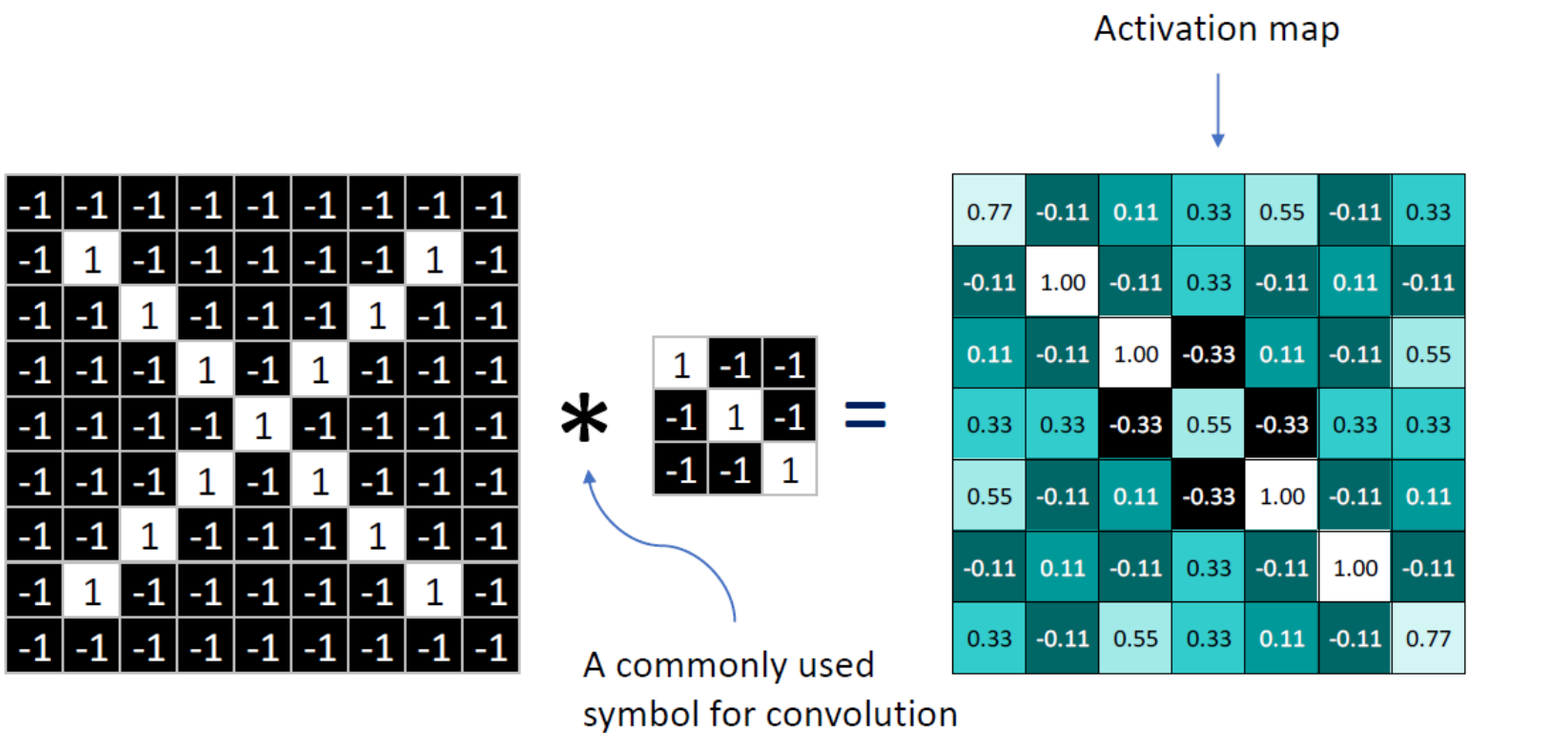
Lab 2

- Align the filter and the image patch.
- Multiply each image pixel by the corresponding filter pixel.
- Add the result.
- Divide by the size of the filter. (Implementation differs!)

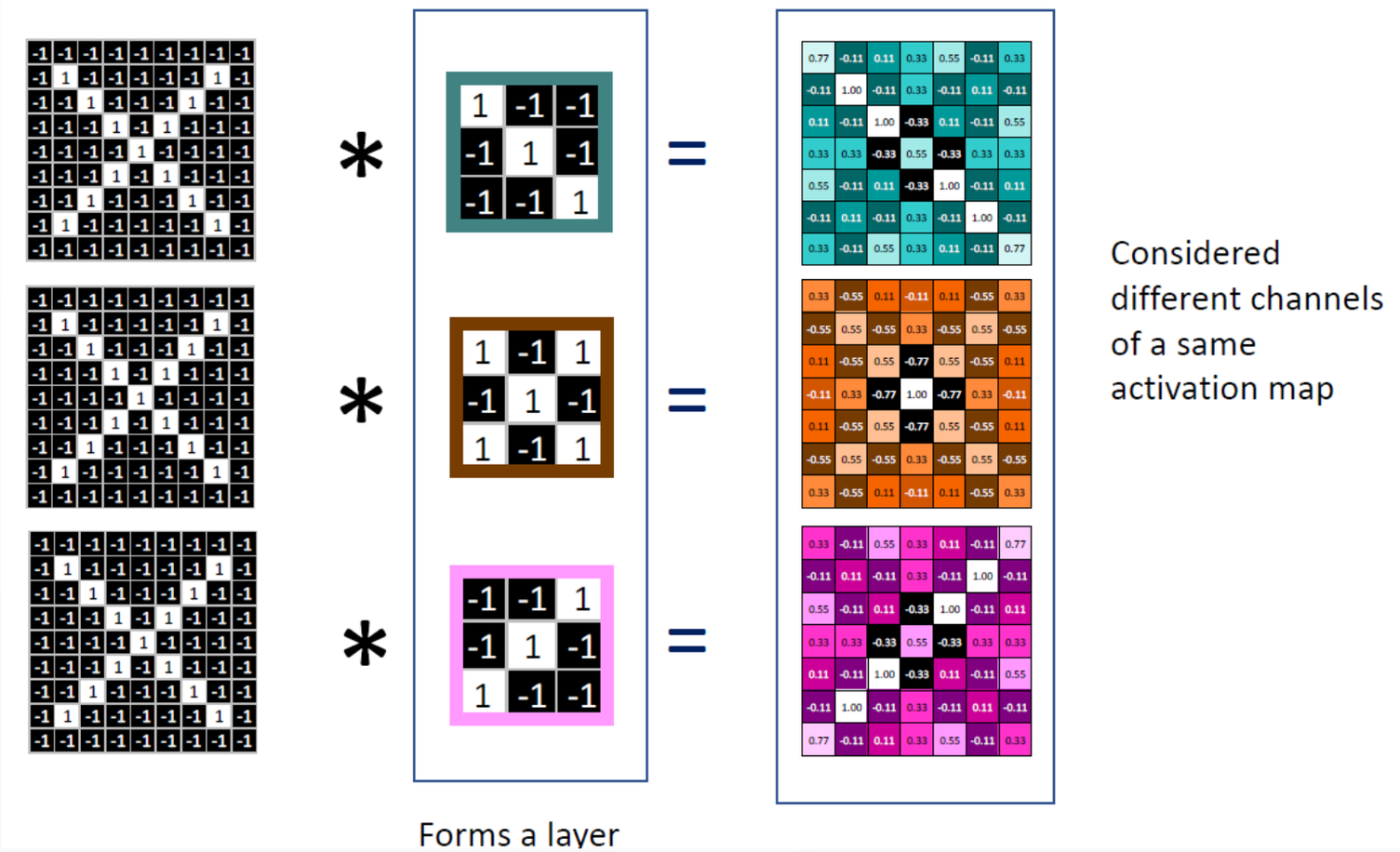


Lab 2

- Activation Map

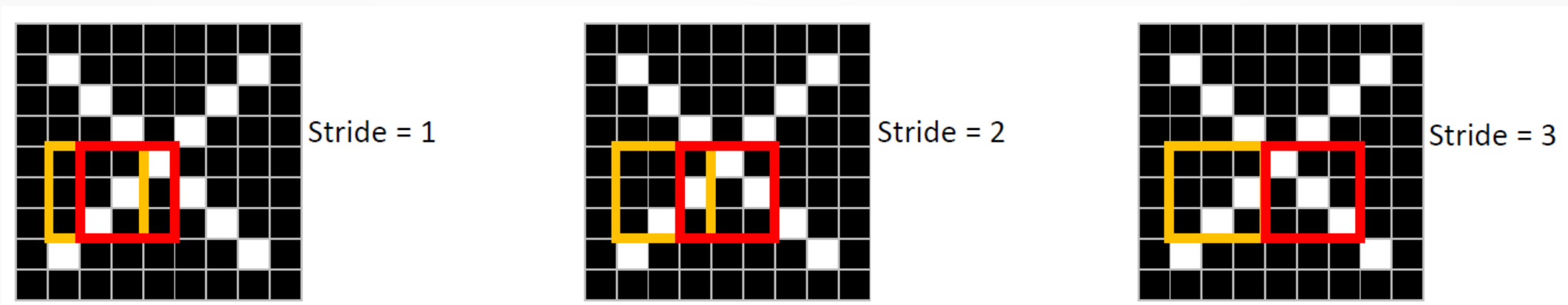


Lab 2


Considered different channels of a same activation map
Forms a layer

Lab 2

- Stride is the distance between two consecutive image patches used during convolution.
- Stride is measured in terms of pixels.
- If stride < patch-width, patches overlap.
- If stride >= patch-width, patches do not overlap.



Lab 2

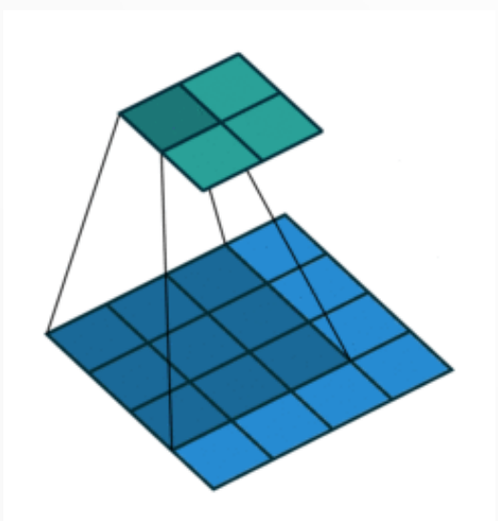
Pad additional pixels on the boundary of the input to achieve a specific output size.

Ways of padding:

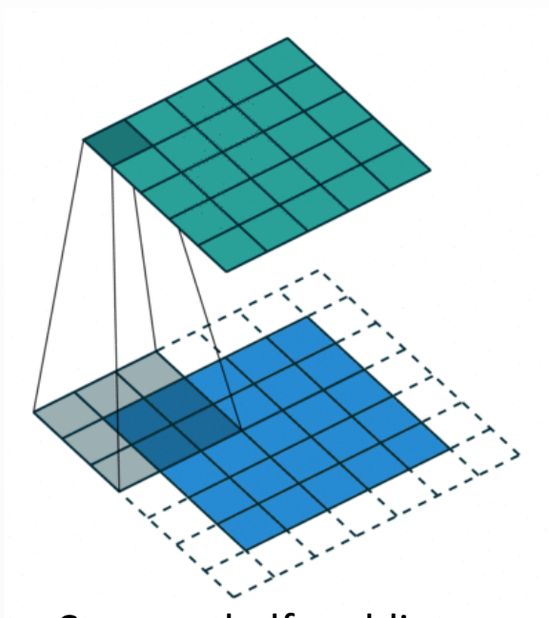
- Pad pixels with value equal to zero
- Repeat the boarder pixels
- Reflect the image around the boarder

Lab 2

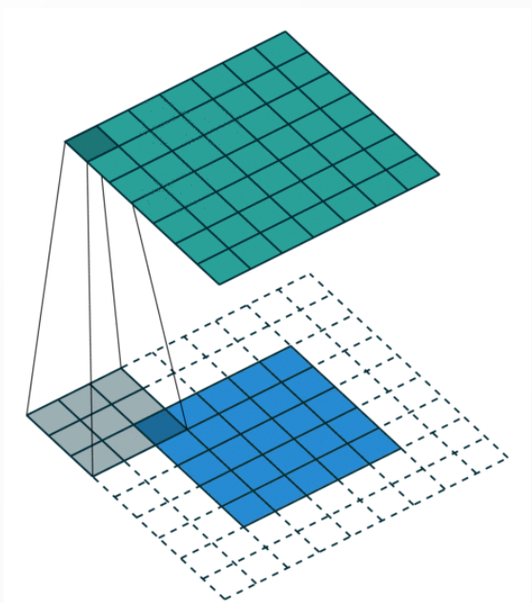
Padding



Valid or no padding
output size < input size



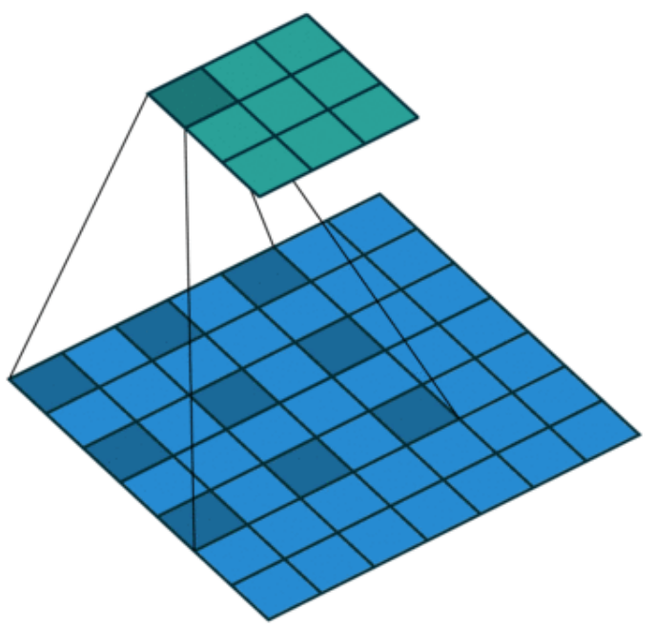
Same or half padding
output size = input size



Full padding
output size > input size

Lab 2

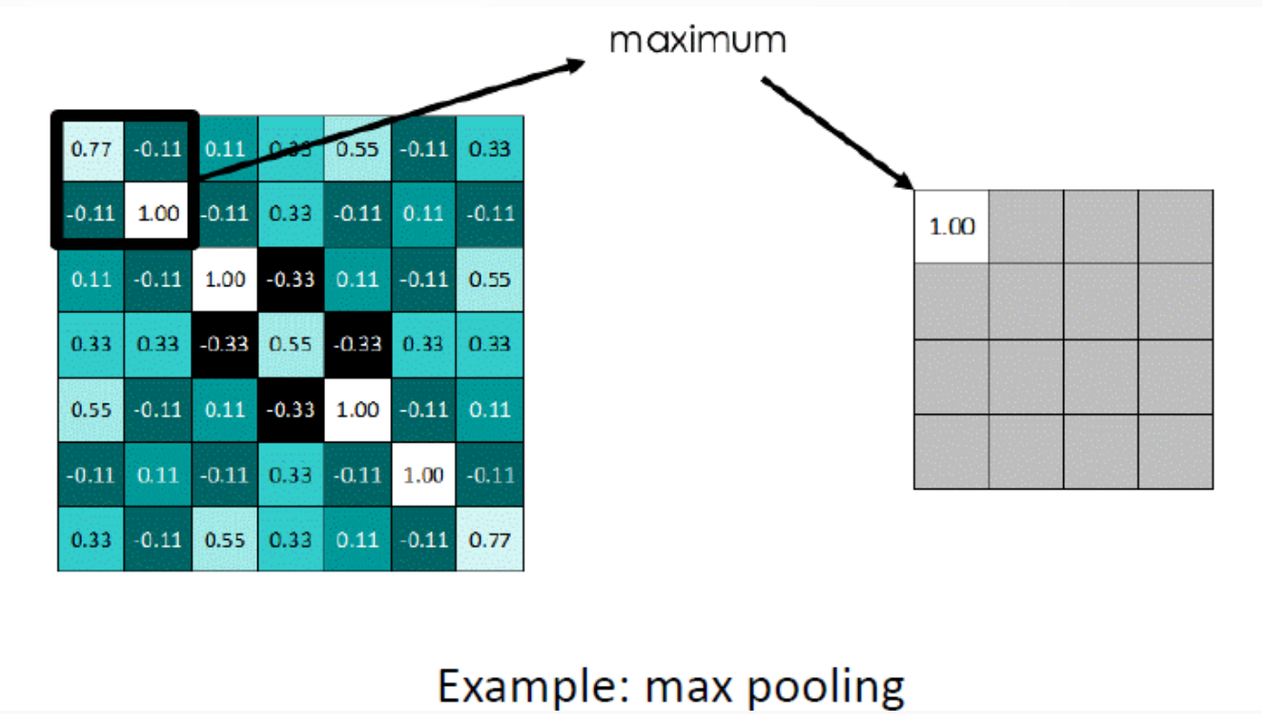
- The extent of the local region connected to one hidden unit is known as receptive field.
- To increases the receptive field without increasing the number of parameters, use dilation.



Dilation = 2

Lab 2

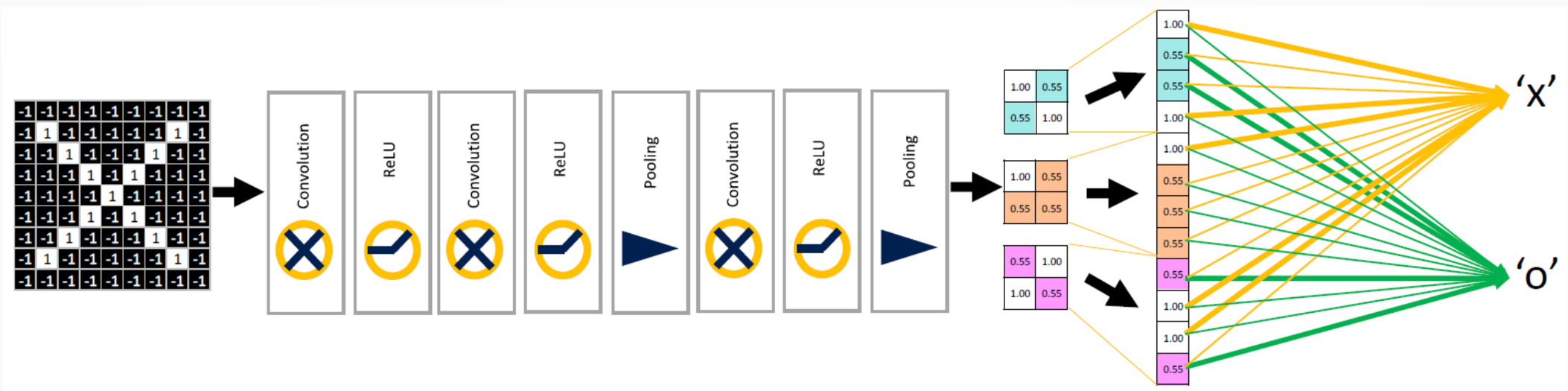
- Pool hidden units in a non overlapping neighborhood in a single channel.
- Pooling functions: max, min, mean, weighted average, etc.
- Pooling introduces invariance to local translations.



Example: max pooling

Lab 2

- In summary



Lab 2

```
from tensorflow.keras import layers, models
model = models.Sequential()
#define filters and convolutional layers here
model.add(layers.Conv2D(filters=16, kernel_size=(3, 3), activation='relu',
input_shape=(28, 28, 1)))
#Add a maxpooling layer
model.add(layers.MaxPooling2D(pool_size=(2, 2)))
#Flatten the output and give it to a fully connected layer
model.add(layers.Flatten())
#one hidden layer maps the flattened neurons to output
model.add(layers.Dense(10, activation='softmax’))
model.compile(optimizer='adam', loss='categorical_crossentropy',
metrics=['accuracy'])
```

Lab 2

- MNIST dataset: a large database of handwritten digits that is commonly used for training various image processing systems.
- Images are 28*28 pixels

```
from tensorflow.keras.datasets import mnist
(X_train, y_train), (X_test, y_test) = mnist.load_data()
```

Lab 2

- Convert the class labels into a one-hot encoding.

- I suggest using the following:

```
from tensorflow.keras.utils import to_categorical
y_train_encoded = to_categorical(y_train)
y_test_encoded = to_categorical(y_test)
```

Lab 2

- You will solve an MNIST classification task using CNN.

- Use a ReLU activation instead of a sigmoid one.
- In your pooling layers, use the MAX() function instead of the arithmetic mean.
- For classification, use a single dense layer followed by a softmax layer.

Lab 2

```
model = models.Sequential()
#Define filters and convolutional layers here
model.add(layers.Conv2D(filters=16, kernel_size=(3, 3), activation='relu',
input_shape=(28, 28, 1)))
#Add a maxpooling layer
model.add(layers.MaxPooling2D(pool_size=(2, 2)))
#Flatten the output and give it to a fully connected layer
model.add(layers.Flatten())
#One hidden layer maps the flattened neurons to output
model.add(layers.Dense(10, activation='softmax’))
model.compile(optimizer=keras.optimizers.Adam(learning_rate=learning_
rate), loss='categorical_crossentropy', metrics=['accuracy'])
```

Lab 2

- Use stratified 5-fold cross validation to test the performance of the models for at least the following parameters:
  - Number of Filters=[16, 32]
  - Learning rate = [0.001, 0.01]

- Once you have determined the best design, train the CNN one more time on the entire training set.
- Report your performance on the out-of-sample test set.

Lab 2

- You have 60,000 training images with 28*28 features, so your training will take a lot of time.