Convolutional Neural Network

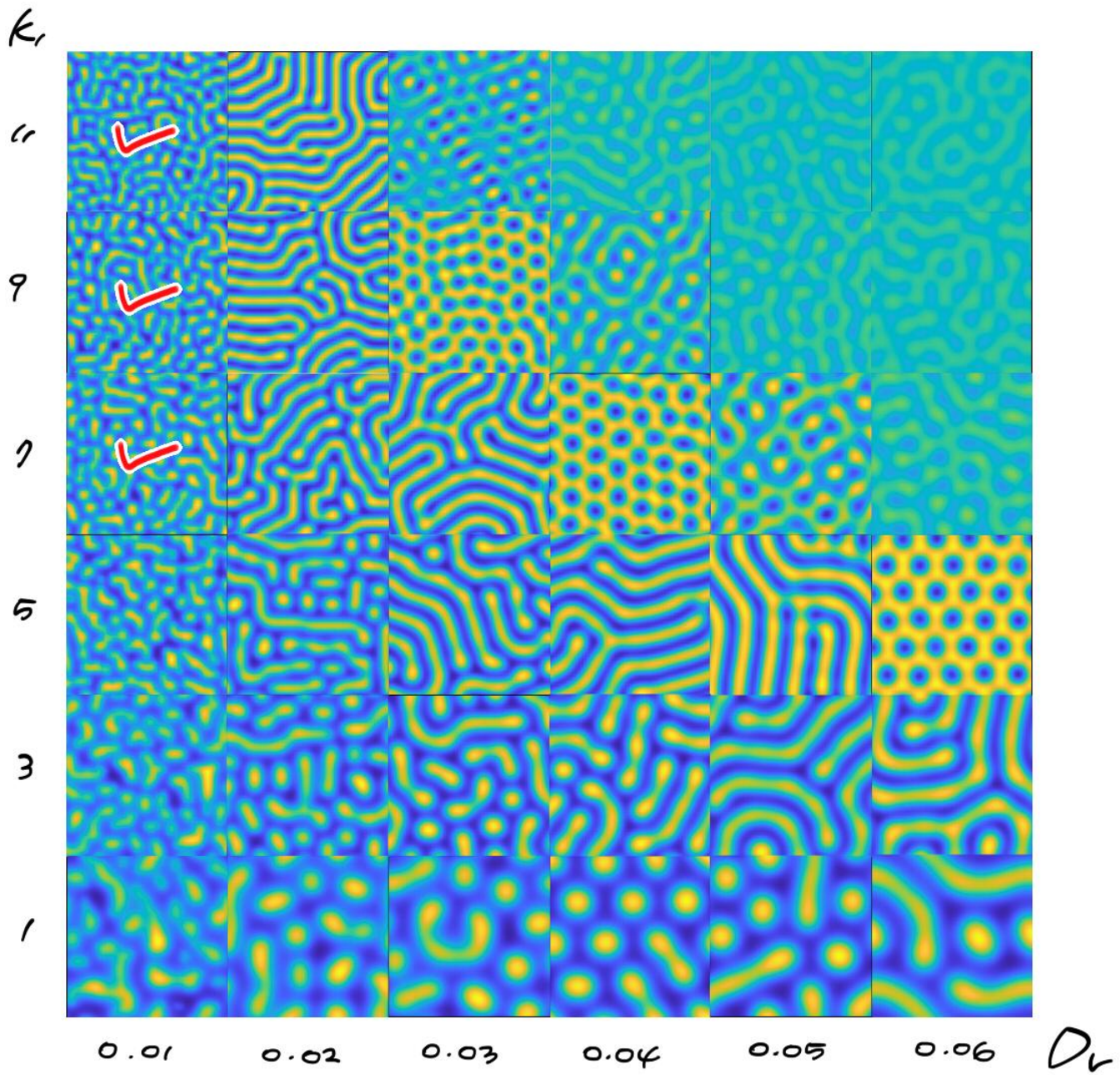
2017010698
수학과 오서영

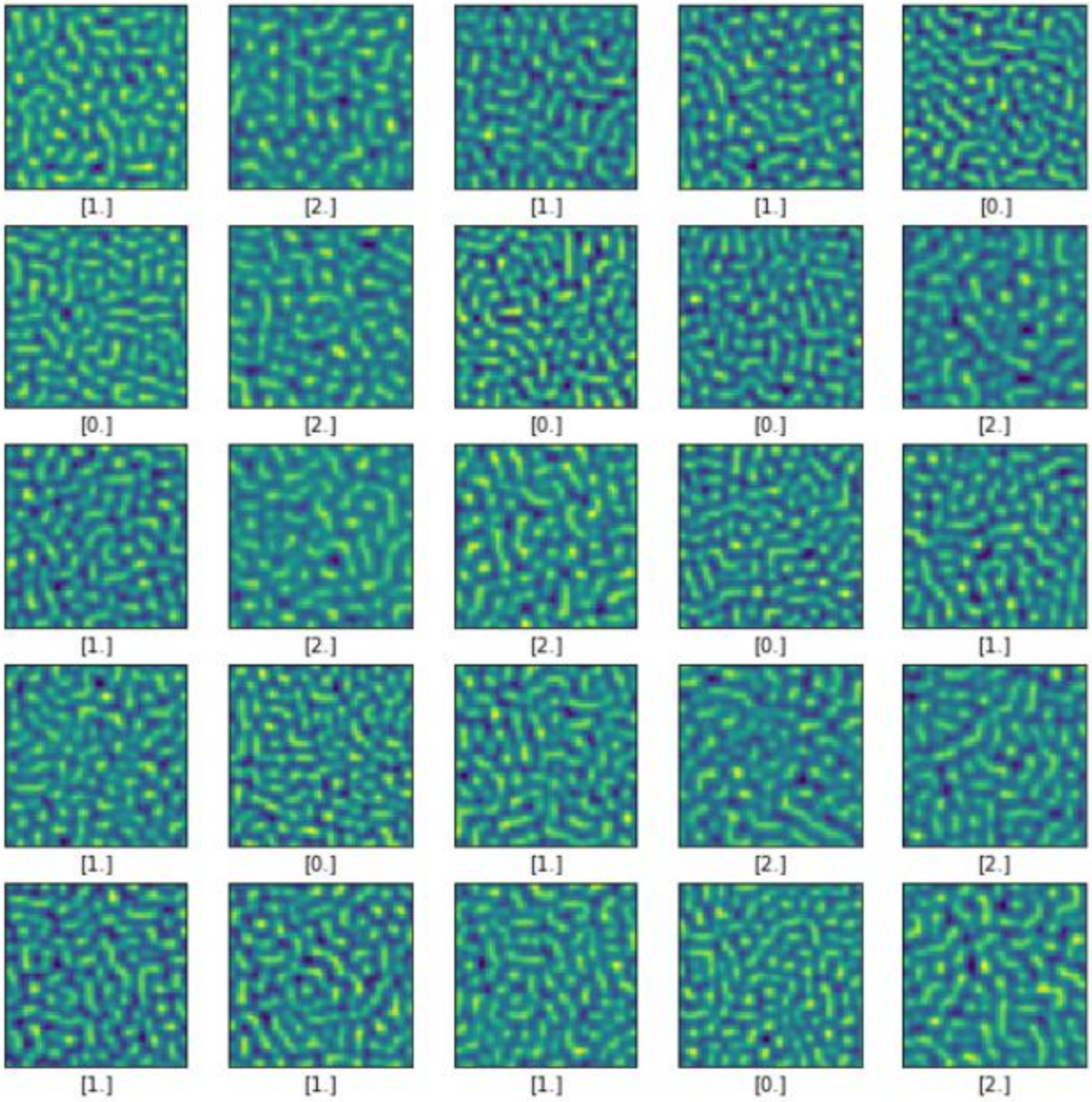
◦ CNN ?

The name “convolutional neural network”
indicates that the network employs
a mathematical operation
called convolution.

Convolution is a specialized kind of
linear operation.
Convolutional networks are simply
neural networks
that use convolution in place of
general matrix multiplication.

most commonly applied to
analyzing visual imagery


k_r
11
9
7
5
3
1
0.01
0.02
0.03
0.04
0.05
0.06
D_r


[1.] [2.] [1.] [1.] [0.]
[0.] [2.] [0.] [0.] [2.]
[1.] [2.] [2.] [0.] [1.]
[1.] [0.] [1.] [2.] [2.]
[1.] [1.] [1.] [0.] [2.]

```
x_train = x_train.reshape(x_train.shape[0], 64, 64, 1)
x_test = x_test.reshape(x_test.shape[0], 64, 64, 1)
batch_size = 128
num_classes = 3
epochs = 30
```

```
model = Sequential()
model.add(Conv2D(32, kernel_size=(5, 5), strides=(1, 1), padding='same',
                 activation='relu',
                 input_shape=(64,64,1)))
model.add(MaxPooling2D(pool_size=(2, 2), strides=(2, 2)))
model.add(Conv2D(64, (2, 2), activation='relu', padding='same'))
model.add(MaxPooling2D(pool_size=(2, 2)))
# model.add(Dropout(0.25))
model.add(Flatten())
model.add(Dense(1000, activation='relu'))
# model.add(Dropout(0.5))
model.add(Dense(num_classes, activation='softmax'))
model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 64, 64, 32)        832
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 32, 32, 32)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 32, 32, 64)        8256
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 16, 16, 64)        0
_________________________________________________________________
flatten_1 (Flatten)          (None, 16384)             0
_________________________________________________________________
dense_1 (Dense)              (None, 1000)              16385000
_________________________________________________________________
dense_2 (Dense)              (None, 3)                 3003
=================================================================
Total params: 16,397,091
Trainable params: 16,397,091
Non-trainable params: 0
```

```
Train on 1200 samples, validate on 300 samples
Epoch 1/30
1200/1200 [==============================] - 7s 6ms/step - loss: 1.7503 - accuracy: 0.3533 - val_loss: 1.1068 - val_accuracy: 0.3100
Epoch 2/30
1200/1200 [==============================] - 7s 5ms/step - loss: 1.1014 - accuracy: 0.3442 - val_loss: 1.1085 - val_accuracy: 0.3100
Epoch 3/30
1200/1200 [==============================] - 6s 5ms/step - loss: 1.1008 - accuracy: 0.3300 - val_loss: 1.0957 - val_accuracy: 0.3300
Epoch 4/30
1200/1200 [==============================] - 7s 6ms/step - loss: 1.0929 - accuracy: 0.3792 - val_loss: 1.0914 - val_accuracy: 0.3100
Epoch 5/30
1200/1200 [==============================] - 7s 6ms/step - loss: 1.0849 - accuracy: 0.3692 - val_loss: 1.0816 - val_accuracy: 0.4833
Epoch 6/30
1200/1200 [==============================] - 7s 6ms/step - loss: 1.0716 - accuracy: 0.4158 - val_loss: 1.0843 - val_accuracy: 0.3100
Epoch 7/30
```

```
1200/1200 [==============================] - 7s 6ms/step - loss: 0.5250 - accuracy: 0.8558 - val_loss: 0.6705 - val_accuracy: 0.6633
Epoch 21/30
1200/1200 [==============================] - 7s 6ms/step - loss: 0.4683 - accuracy: 0.8575 - val_loss: 0.9998 - val_accuracy: 0.4167
Epoch 22/30
1200/1200 [==============================] - 7s 6ms/step - loss: 0.6267 - accuracy: 0.6650 - val_loss: 0.7922 - val_accuracy: 0.5767
Epoch 23/30
1200/1200 [==============================] - 7s 6ms/step - loss: 0.5339 - accuracy: 0.7650 - val_loss: 0.6523 - val_accuracy: 0.7367
Epoch 24/30
1200/1200 [==============================] - 7s 6ms/step - loss: 0.3624 - accuracy: 0.9433 - val_loss: 0.5982 - val_accuracy: 0.7300
Epoch 25/30
1200/1200 [==============================] - 7s 6ms/step - loss: 0.3089 - accuracy: 0.9525 - val_loss: 0.6099 - val_accuracy: 0.7167
Epoch 26/30
1200/1200 [==============================] - 7s 6ms/step - loss: 0.3093 - accuracy: 0.9158 - val_loss: 0.6993 - val_accuracy: 0.6533
Epoch 27/30
1200/1200 [==============================] - 7s 6ms/step - loss: 0.2922 - accuracy: 0.9167 - val_loss: 0.7157 - val_accuracy: 0.6433
Epoch 28/30
1200/1200 [==============================] - 7s 6ms/step - loss: 0.2685 - accuracy: 0.9292 - val_loss: 0.6701 - val_accuracy: 0.6700
Epoch 29/30
1200/1200 [==============================] - 7s 6ms/step - loss: 0.2315 - accuracy: 0.9467 - val_loss: 0.6161 - val_accuracy: 0.7200
Epoch 30/30
1200/1200 [==============================] - 7s 6ms/step - loss: 0.2009 - accuracy: 0.9608 - val_loss: 0.4719 - val_accuracy: 0.8000
```

```
score = model.evaluate(x_test, y_test, verbose=0)
print('Test loss:', score[0])
print('Test accuracy:', score[1])
```

```
Test loss: 0.4718879250685374
Test accuracy: 0.800000011920929
```