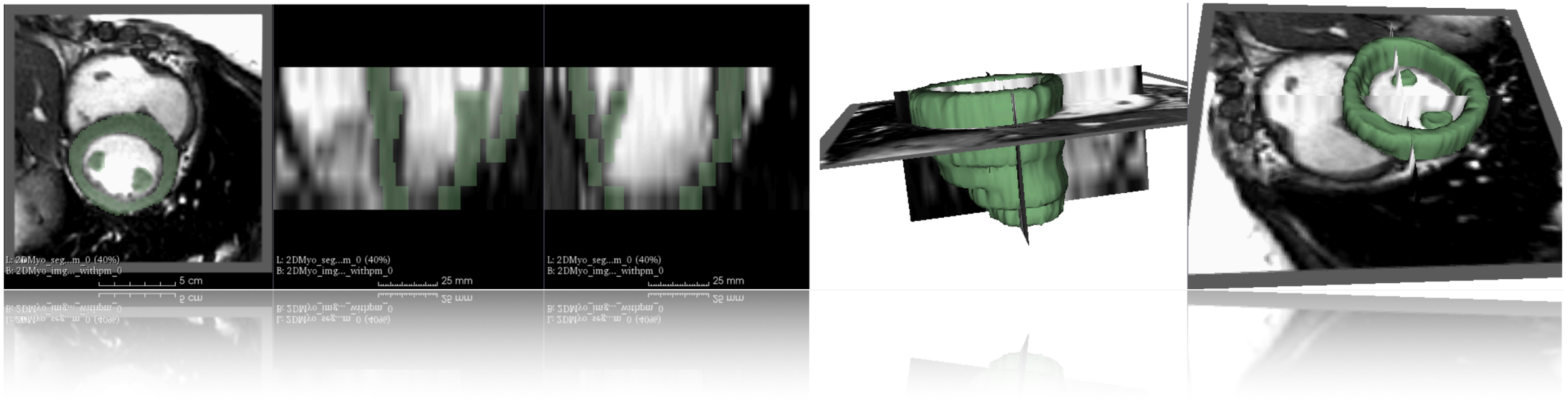


Departamento de Física Médica - Centro atómico Bariloche - IB

Una introducción a Deep Learning con Keras

Ariel Hernán Curiale

https://blog.keras.io

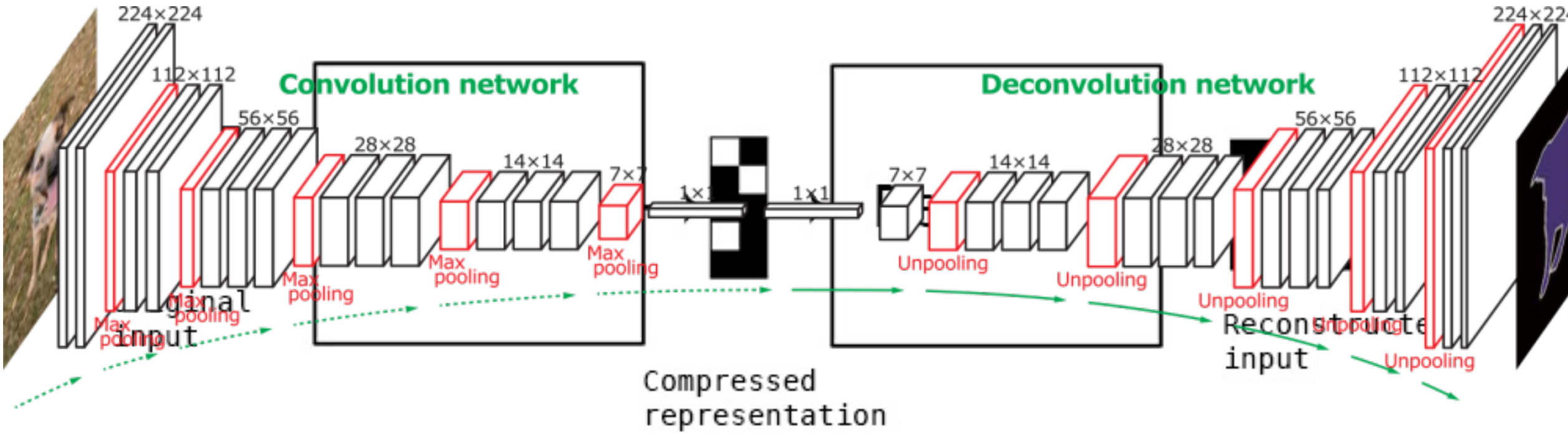
CONICET

UNCUYO
UNIVERSIDAD NACIONAL DE CUYO

Instituto Balseiro

CNEA

Eliminación de ruido



https://blog.keras.io/building-autoencoders-in-keras.html

Configuración

- Keras 1.2

```
ariel@Goku:~/RedesNeuronales/Practicas/Keras$ cat ~/.keras/keras.json
{
    "epsilon": 1e-07,
    "image_dim_ordering": "tf",
    "backend": "tensorflow",
    "floatx": "float32"
}
```

- Keras 2

```
ariel@Goku:~/RedesNeuronales/Practicas/Keras$ cat ~/.keras/keras.json
{
    "epsilon": 1e-07,
    "image_data_format": “channels_last",
    "backend": "tensorflow",
    "floatx": "float32"
}
```

- `image_data_format`: String, either `"channels_last"` or `"channels_first"`. It specifies which data format convention Keras will follow. (`keras.backend.image_data_format()` returns it.)
- For 2D data (e.g. image), `"channels_last"` assumes `(rows, cols, channels)` while `"channels_first"` assumes `(channels, rows, cols)`.
- For 3D data, `"channels_last"` assumes `(conv_dim1, conv_dim2, conv_dim3, channels)` while `"channels_first"` assumes `(channels, conv_dim1, conv_dim2, conv_dim3)`.
- `epsilon`: Float, a numeric fuzzing constant used to avoid dividing by zero in some operations.
- `floatx`: String, `"float16"`, `"float32"`, or `"float64"`. Default float precision.
- `backend`: String, `"tensorflow"`, `"theano"`, or `"cntk"`.

```
ariel@Goku:~/RedesNeuronales/Practicas/Keras$ ipython
Python 3.6.4 | packaged by conda-forge | (default, Dec 23 2017, 16:54:01)
Type 'copyright', 'credits' or 'license' for more information
IPython 6.2.1 -- An enhanced Interactive Python. Type '?' for help.

In [1]: import keras
Using TensorFlow backend.

In [2]: keras.__version__
Out[2]: ‘2.1.5'
```

Eliminación de ruido

- Creamos la red

```
from keras.layers import Input, Conv2D, MaxPooling2D, UpSampling2D
from keras.models import Model
input_layer = Input(shape=(28, 28, 1))  # adapt this if it is using `channels_first`
image data format
nfilters = 32
kernel_size = [3,3]

layer = Conv2D(nfilters, kernel_size, use_bias=True,
               activation='relu', padding='same')(input_layer)
layer = MaxPooling2D(pool_size=(2, 2), padding='same')(layer)
layer = Conv2D(nfilters, kernel_size, use_bias=True,
               activation='relu', padding='same')(layer)
encoded = MaxPooling2D(pool_size=(2, 2), padding='same')(layer)
# at this point the representation is (7, 7, 32)
layer = Conv2D(nfilters, kernel_size, use_bias=True,
               activation='relu', padding='same')(encoded)
layer = UpSampling2D(size=(2, 2))(layer)
layer = Conv2D(nfilters, kernel_size, use_bias=True,
               activation='relu', padding='same')(layer)
layer = UpSampling2D(size=(2, 2))(layer)
decoded = Conv2D(1, kernel_size, use_bias=True,
               activation='sigmoid', padding='same')(layer)

model = Model(input_layer, decoded)
```

Eliminación de ruido

- Elijamos una estrategia de optimización y una métrica o función a minimizar

```
model.compile(optimizer='adadelta', loss='binary_crossentropy',
              metrics='accuracy')
```

- Mostramos un resumen del modelo

```
In [2]: model.summary()
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         (None, 28, 28, 1)         0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 28, 28, 32)        320
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 14, 14, 32)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 14, 14, 32)        9248
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 7, 7, 32)          0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 7, 7, 32)          9248
_________________________________________________________________
up_sampling2d_1 (UpSampling2 (None, 14, 14, 32)        0
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 14, 14, 32)        9248
_________________________________________________________________
up_sampling2d_2 (UpSampling2 (None, 28, 28, 32)        0
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 28, 28, 1)         289
=================================================================
Total params: 28,353
Trainable params: 28,353
Non-trainable params: 0
_________________________________________________________________
```

Eliminación de ruido

- Cargamos los datos

```
import numpy as np
tmp = np.load('mnist_database.npz')
x_train, y_train = tmp['x_train'], tmp['y_train']
x_test, y_test = tmp['x_test'], tmp['y_test']
x_train = x_train.astype('float32') / 255.
x_test = x_test.astype('float32') / 255.
print(x_train.shape)
print(x_test.shape)

x_train = np.reshape(x_train, (len(x_train), 28, 28, 1))  # adapt this if it is using
`channels_first` image data format
x_test = np.reshape(x_test, (len(x_test), 28, 28, 1))  # adapt this if it is using
`channels_first` image data format

noise_factor = 0.5
x_train_noisy = x_train + noise_factor * np.random.normal(loc=0.0, scale=1.0,
size=x_train.shape)
x_test_noisy = x_test + noise_factor * np.random.normal(loc=0.0, scale=1.0,
size=x_test.shape)
x_train_noisy = np.clip(x_train_noisy, 0., 1.)
x_test_noisy = np.clip(x_test_noisy, 0., 1.)
```

Eliminación de ruido

- Entrenamos la red neuronal

```
history = model.fit(x_train_noisy, x_train,
                    epochs=5,
                    batch_size=128,
                    shuffle=True,
                    verbose=1,
                    validation_data=(x_test_noisy, x_test))
```

- Predicción / eliminación de ruido

```
x_test_denoisy = model.predict(x_test_noisy, verbose=1)
```

Eliminación de ruido

- Visualizamos los resultados

```
# Matplotlib
import matplotlib.pyplot as plt
f, ax = plt.subplots(1,3)
ax[0].imshow(x_test[0, ..., 0], 'gray')
ax[1].imshow(x_test_noisy[0, ..., 0], 'gray')
ax[2].imshow(x_test_denoisy[0, ..., 0], 'gray')

plt.figure()
plt.title('Error on training performance')
plt.plot(history.history['loss'], '-b', label='loss')
plt.plot(history.history['val_loss'], '-r',
label='val_loss')
plt.xlabel('Number of epochs')
plt.ylabel('Error loss')
plt.legend()
```