Lecture 08

Deep Learning 03

Convolutional encoder-decoders

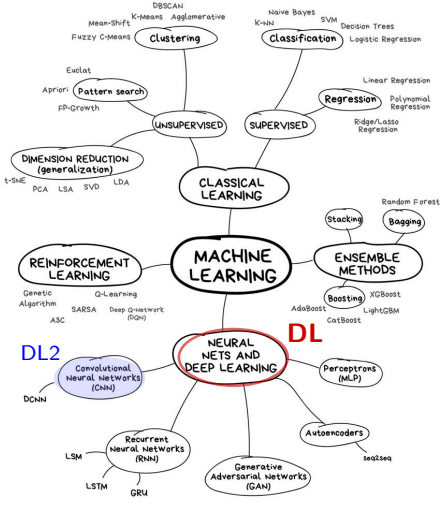
2026-04-24

Sébastien Valade

VNIVERSIDAD NACIONAL
AVPENOMA DE
MEXICO

Introduction

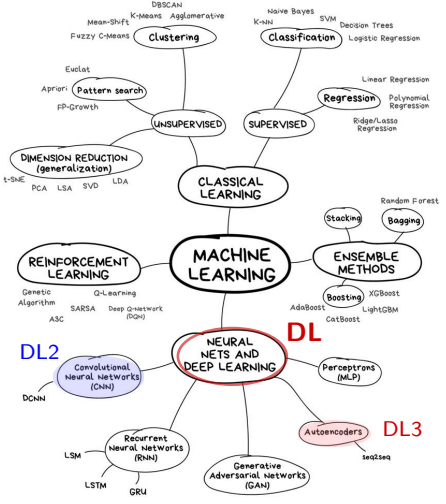
Last week: CNN part-1 *(CNNs for image classification)* - DL2



source

Introduction

Last week: CNN part-1 *(CNNs for image classification)* - DL2
This week: CNN part-2 *(CNNs for semantic segmentation)* - DL3



source

Main Computer Vision tasks: classification, detection, segmentation

Classification

- ✓ Presence
- × Location
- × Count
- × Size
- × Shape

Credit: cloudfactory

Main Computer Vision tasks: classification, detection, segmentation

Classification

- ✓ Presence
- × Location
- × Count
- × Size
- × Shape

Object Detection

- ✓ Presence
- ✓ Location
- ✓ Count
- × Size
- × Shape

Credit: cloudfactory

Main Computer Vision tasks: classification, detection, segmentation

Classification

- ✓ Presence
- × Location
- × Count
- × Size
- × Shape

Object Detection

- ✓ Presence
- ✓ Location
- ✓ Count
- × Size
- × Shape

Semantic Segmentation

- ✓ Presence
- ✓ Location
- × Count
- ! Size
- ! Shape

Credit: cloudfactory

Main Computer Vision tasks: classification, detection, segmentation

Classification	Object Detection	Semantic Segmentation	Instance Segmentation
			
✓ Presence	✓ Presence	✓ Presence	✓ Presence
× Location	✓ Location	✓ Location	✓ Location
× Count	✓ Count	× Count	✓ Count
× Size	× Size	! Size	✓ Size
× Shape	× Shape	! Shape	✓ Shape

Credit: cloudfactory

Auto-encoders (AE): architecture for learning latent representations

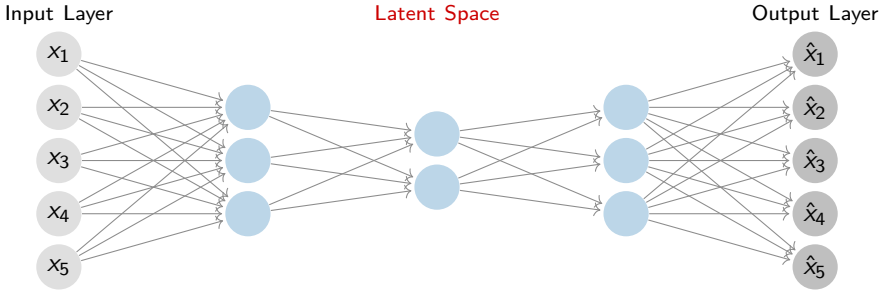
⇒ first introduced in the 1980s[1] as neural networks trained to reconstruct their input

⇒ *unsupervised learning task that learns by minimizing reconstruction error*

⇒ network learns dense representations of the input data: **latent representation**, **latent space**, or **codings**

⇒ early use was dimensionality reduction & compression

NB: if the autoencoder uses only linear activations and the cost function is the mean squared error (MSE), then it ends up performing principal component analysis (PCA)

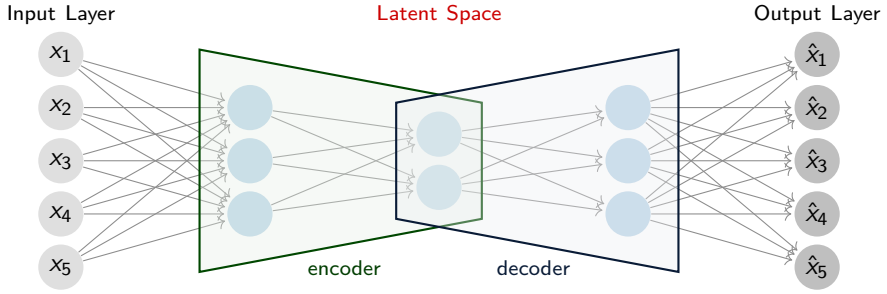


[1] *Rumelhart, Hinton, Williams (1986) PDP Vol.1, "Learning Internal Representations by Error Propagation"*

Auto-encoders (AE): architecture for learning latent representations

⇒ AE architecture: **encoder–decoder** pattern

- **encoder**
  ⇒ compresses input to low-dimensional latent space
- **latent space** (bottleneck)
  ⇒ low-dimensional representation of the input *(captures the most salient features of the input)*
- **decoder**
  ⇒ reconstructs input from latent space



Encoder–decoder pattern: general architecture used for many tasks

⇒ 1980+: encoder-decoder for *dimensionality reduction, compression* (first AE)[1]

⇒ 2014+: encoder-decoder for *natural language processing* NLP (*seq2seq model*[2])

⇒ 2015+: encoder-decoder for *image processing* (segmentation[3,4,5,6], translation[7], denoising, generation)

***Note**: the term "auto-encoder" originally refers to a reconstruction task (input = output), but the term is often used loosely to mean **encoder-decoder architecture** even when outputs differ.*

[1] *Rumelhart et al. (1986) PDP Vol.1, "Learning Internal Representations by Error Propagation"*

[2] *Sutskever et al. (2014) "Sequence to Sequence Learning with Neural Networks" → a.k.a. seq2seq*

[3] *Long et al. (2015) "Fully Convolutional Networks for Semantic Segmentation" → a.k.a. FCN*

[4] *Noh et al. (2015) "Learning Deconvolution Network for Semantic Segmentation" → a.k.a. DeconvNet*

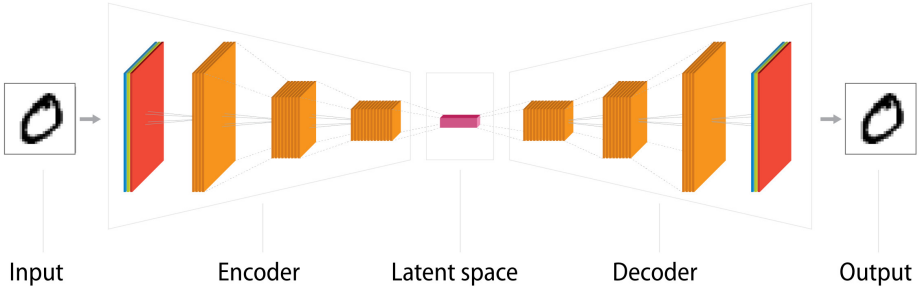
[5] *Badrinarayanan et al. (2015) "SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation" → a.k.a. SegNet*

[6] *Ronneberger et al. (2015) "U-Net: Convolutional Networks for Biomedical Image Segmentation" → a.k.a. U-Net*

[7] *Valade et al. (2019) "Towards Global Volcano Monitoring Using Multisensor Sentinel Missions and Artificial Intelligence"*

Convolutional encoder-decoder

- **encoder**
  - ⇒ **downsampling** using convolution + pooling
  - ⇒ *reduce spatial dimensionality* (downscale width/height) & *increase depth* (number of feature maps)
- **decoder**
  - ⇒ **upsampling** using transpose convolution + unpooling/upsampling
  - ⇒ *increase spatial dimensionality* (upscale width/height) & *reduce depth* back to original dimensions



credit: Fiddle

Convolutional encoder-decoder

- **encoder**
  - ⇒ **downsampling** using convolution + pooling
  - ⇒ *reduce spatial dimensionality* (downscale width/height) & *increase depth* (number of feature maps)
- **decoder**
  - ⇒ **upsampling** using transpose convolution + unpooling/upsampling
  - ⇒ *increase spatial dimensionality* (upscale width/height) & *reduce depth* back to original dimensions

```python
# Encoder
inputs = keras.Input(shape=(28, 28, 1))
x = layers.Conv2D(32, 3, activation="relu", strides=2, padding="same")(inputs)
x = layers.Conv2D(64, 3, activation="relu", strides=2, padding="same")(x)
x = layers.Flatten()(x)
x = layers.Dense(16, activation="relu")(x)
z = layers.Dense(latent_dim)(x)
encoder = keras.Model(inputs, z, name="encoder")
```

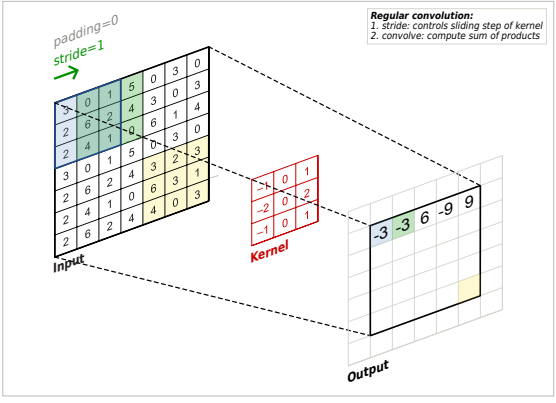
```python
# Decoder
inputs = keras.Input(shape=(latent_dim,))
x = layers.Dense(7 * 7 * 64, activation="relu")(inputs)
x = layers.Reshape((7, 7, 64))(x)
x = layers.Conv2DTranspose(64, 3, activation="relu", strides=2, padding="same")(x)
x = layers.Conv2DTranspose(32, 3, activation="relu", strides=2, padding="same")(x)
outputs = layers.Conv2DTranspose(1, 3, activation="sigmoid", padding="same")(x)
decoder = keras.Model(inputs, outputs, name="decoder")
```

```python
# Autoencoder
inputs = keras.Input(shape=(28, 28, 1))
latents = encoder(inputs)
outputs = decoder(latents)

ae = keras.Model(inputs, outputs, name="ae")
```

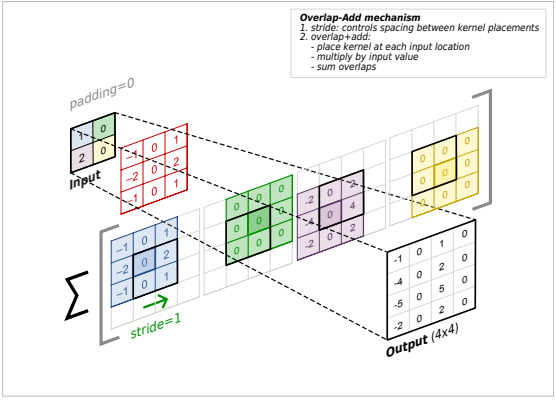
1. convolution vs transpose convolution

Convolution



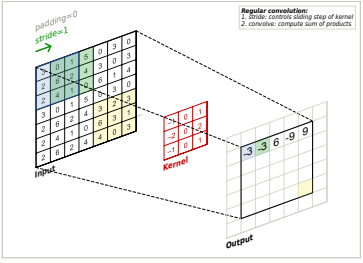
⇒ downscales input
⇒ kernel slides over input (sum of products)

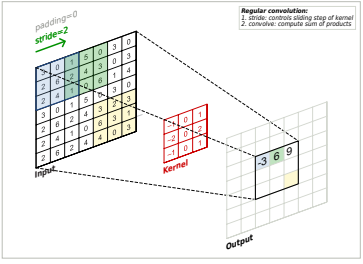
Transposed Convolution



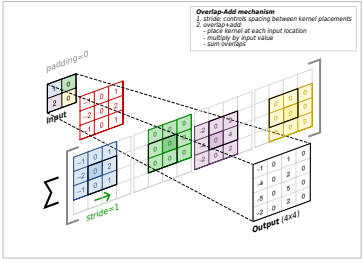
⇒ upscales input
⇒ kernel spreads input (overlap-add mechanism)

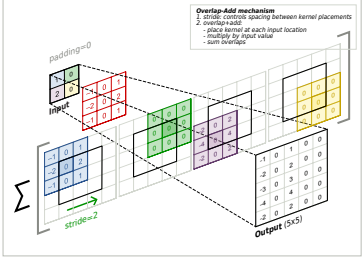
1. convolution vs transpose convolution





⇒ stride controls downscaling factor

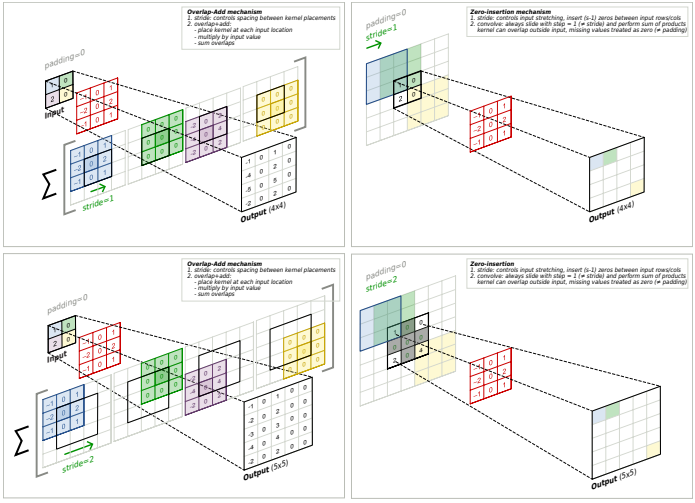




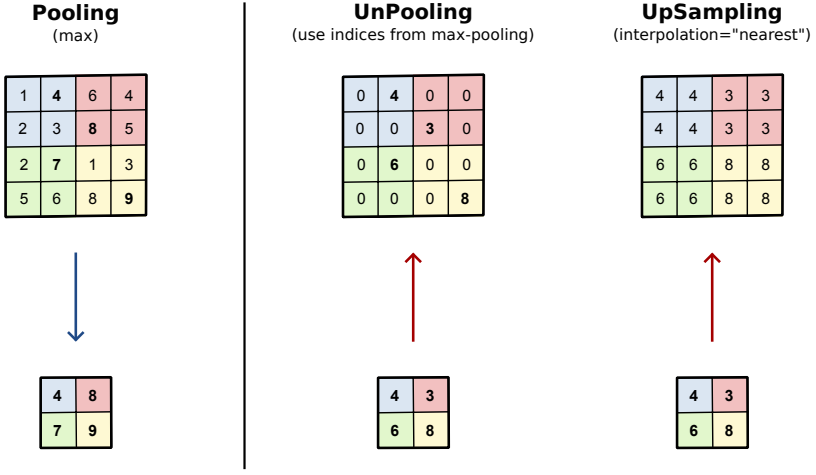
⇒ stride controls upscaling factor

2.2. convolutional encoder-decoder

Note: *there are variants of how the transpose convolution is implemented (e.g., overlap-add mechanism vs. zero-inserted mechanism, aka fractionally strided convolution), but the idea is the same: kernel spreads the input to produce an upscaled output, stride controls the upscaling factor, weights are learned.*

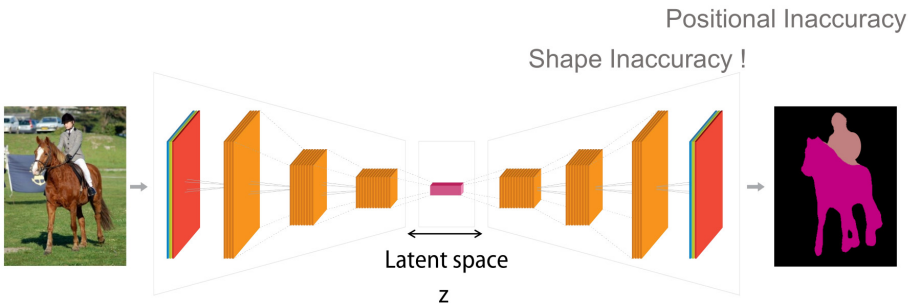


2. pooling vs unpooling/upsampling



U-net architecture: convolutional encoder-decoder with skip connections

Problem *with convolutional encoder-decoders: can lose spatial information (e.g., object boundaries) during downsampling, which can lead to shape/positional inaccuracies in the output.*

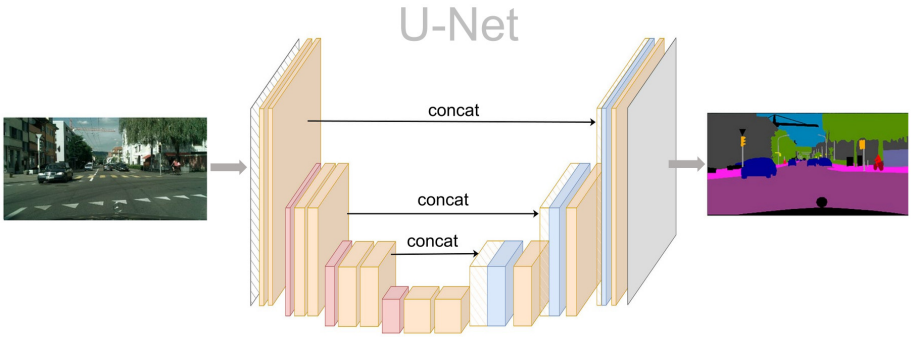


credit: Fiddle

U-net architecture: convolutional encoder-decoder with skip connections

Solution: *add skip connections between encoder and decoder layers to preserve spatial information.*

⇒ *direct links between encoder and decoder layers at the same level*
⇒ *preserve spatial information (e.g., object boundaries) during downsampling*
⇒ *improve shape/positional accuracy in output*



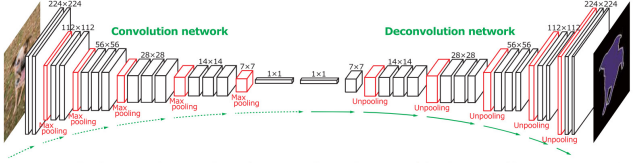
credit: Fiddle

Applications of convolutional encoder-decoders

1. Image segmentation

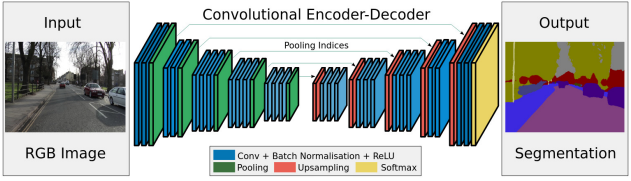
- Noh et al. 2015 (DeconvNet)

  ⇒ *encoder based on VGG-16; bottleneck 2FC; decoder uses unpooling + deconvolution (heavier)*



- Badrinarayanan et al. 2015 (SegNet)

  ⇒ *encoder based on VGG-16; decoder uses pooling indices for unpooling (no learned upsampling; lightert)*

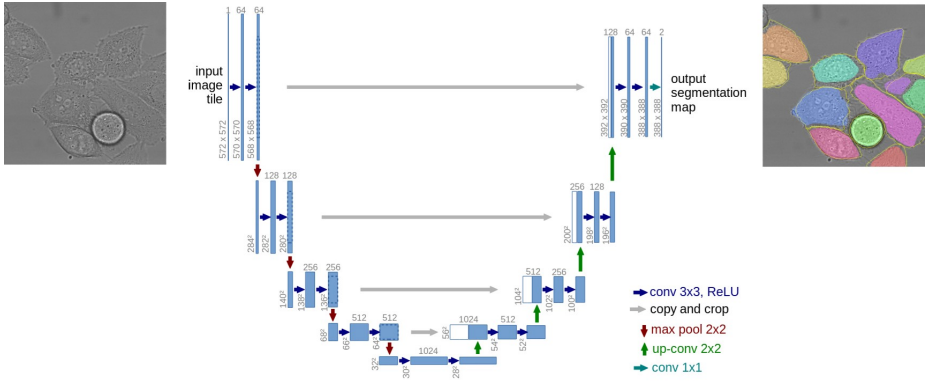


Applications of convolutional encoder-decoders

1. **Image segmentation** (continued)

   - Ronneberger et al. 2015 (U-Net)

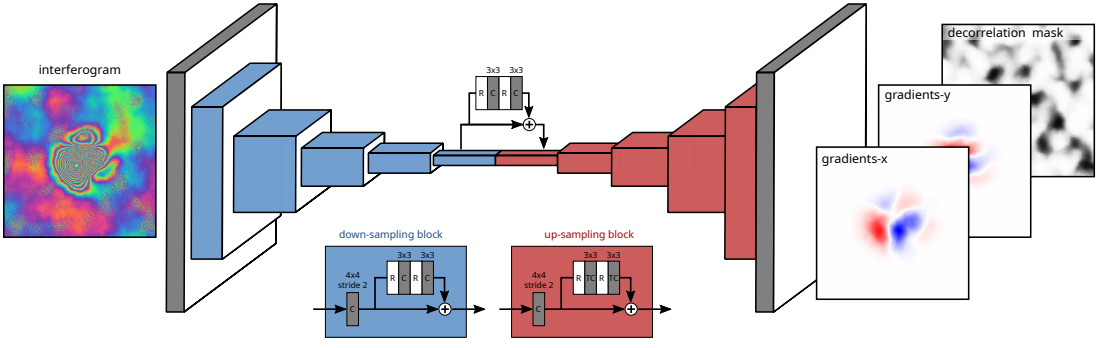
     ⇒ *encoder based on VGG-16; decoder uses unpooling + deconvolution + skip connections (more precise)*



Applications of convolutional encoder-decoders

2. Image "translation"

- Valade et al. 2019
  - ⇒ *encoder-decoder with ResNet inspired blocs*
  - ⇒ *used in real-time in the monitoring system MOUNTS to robustly detect large volcano deformation*

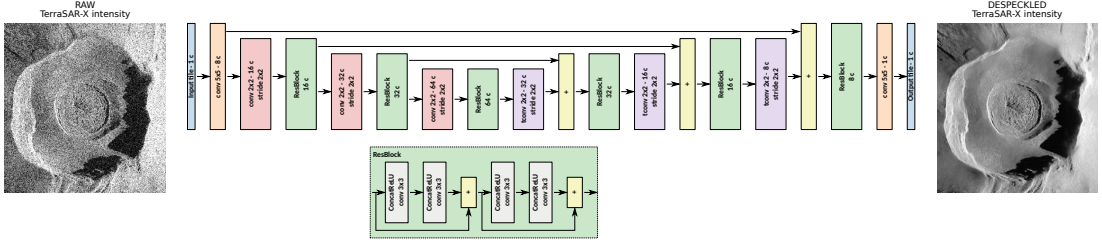


Valade et al. 2019 "Towards Global Volcano Monitoring Using Multisensor Sentinel Missions and Artificial Intelligence: The MOUNTS Monitoring System"

Applications of convolutional encoder-decoders

3. Image filtering

- Valade et al. 2019
  - ⇒ *U-Net architecture with ResNet inspired blocs*
  - ⇒ *used to despeckle TerraSAR-X images*

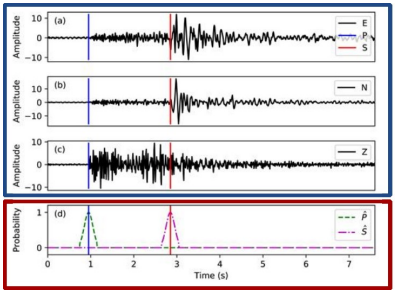


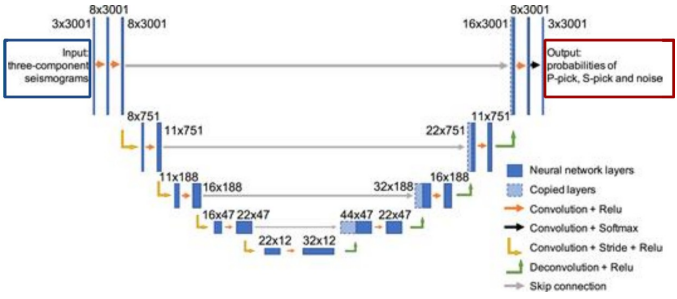
Valade et al. 2023 "Lava dome cycles reveal rise and fall of magma column at Popocatépetl volcano", Nature Communications

Applications of convolutional encoder-decoders

4. Application to 1-D time-series data

- Zhu and Beroza 2019
  - ⇒ *U-net architecture adapted for 1-D seismic data (instead of 2-D images)*
  - ⇒ *used for earthquake P-S phase picking*





Zhu and Beroza 2019 "PhaseNet: A Deep Neural Network for Seismic Phase Picking", Geophysical Journal International

Loss vs metrics

- **Loss**: objective used to train the model (differentiable; minimized by optimizer)
- **Metric**: evaluation measure to report performance (may be non-differentiable; not optimized directly)

⇒ *Why separate roles: losses guide learning; metrics communicate quality to humans*

⇒ *Overlap between roles, example: "Dice" can be a loss (Dice loss) or a metric (Dice score); same idea, different role*

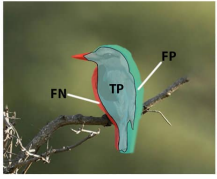
Intersection over Union (IoU)

$\Rightarrow$ IoU $= \frac{\text{area of overlap}}{\text{area of union}}$

$\Rightarrow$ IoU $= \frac{TP}{TP+FP+FN}$

Ground Truth Mask

Predicted Mask



IoU vs Dice

⇒ IoU (Jaccard): $|A \cap B|/|A \cup B|$

⇒ Dice (F1): $2|A \cap B|/(|A| + |B|)$

⇒ relation: $\text{Dice} = 2\,\text{IoU}/(1 + \text{IoU})$

⇒ both used as **metrics**; both can be used as **losses**

⇒ Dice is more forgiving on small objects