

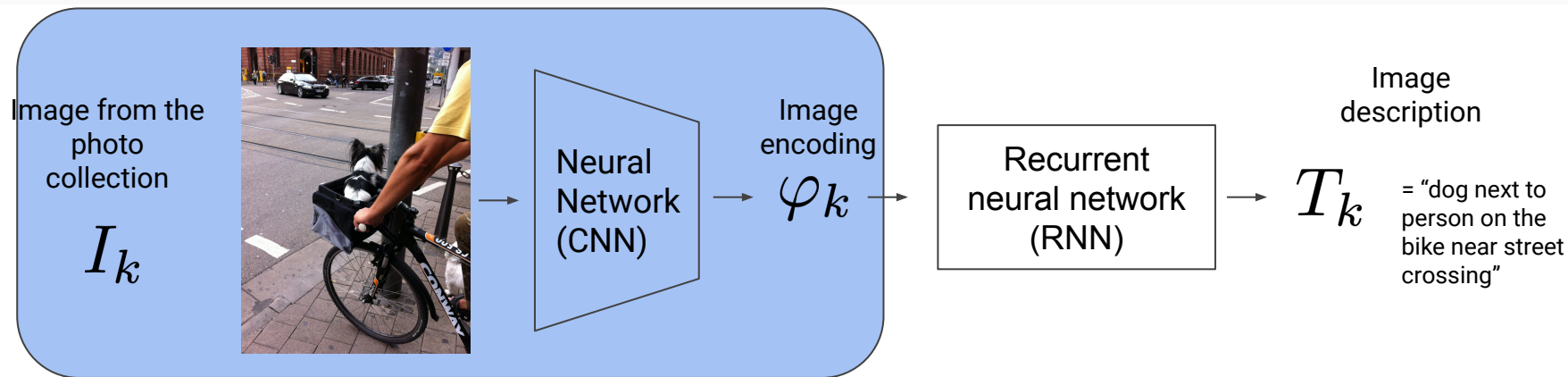
Sapienza Training Camp 2020

Building an Image Search Engine

3 - 5 September, 2020



Roadmap



Define similarity function. Order images according to similarity to the query.

Q

Query: "person walking with a dog on the beach"

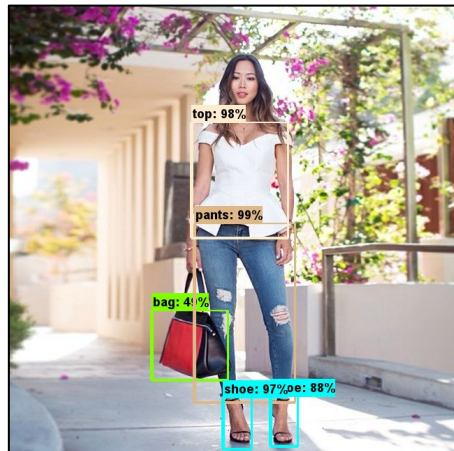


$$\text{sim}(Q, T_1) > \text{sim}(Q, T_2)$$

Agenda for today

- Introduction to Neural Networks
- Convolutional Neural Networks (CNN)
- Modern CNN Architectures

Computer Vision



(a) Image



(b) G.T.

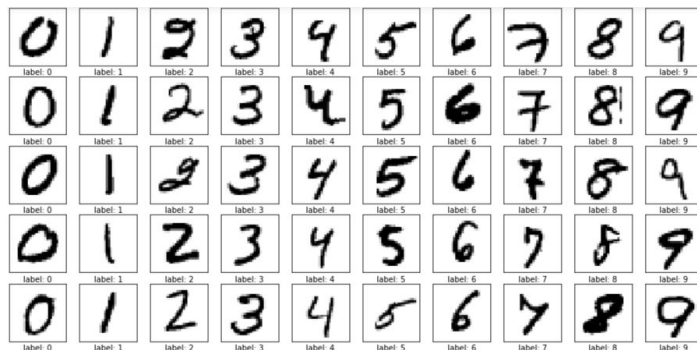


(c) DeepLab-CRF



Neural Networks Recap

- Let's start with a simple neural network from the self-study tutorial
- The task is to recognize handwritten digits



Neural Networks Recap

```
model = tf.keras.Sequential([
    tf.keras.layers.Flatten(input_shape=(28, 28, 1)),
    tf.keras.layers.Dense(200, activation='relu'),
    tf.keras.layers.Dense(60, activation='relu'),
    tf.keras.layers.Dense(10, activation='softmax') # classifying into 10 classes
])

# this configures the training of the model. Keras calls it "compiling" the model.
model.compile(optimizer='adam',
              loss='categorical_crossentropy',
              metrics=['accuracy'])

# train the model
model.fit(training_dataset, ...)
```

Basic model from the self-study course:

<https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist>

Computation within a single neuron

- Basic building block: single neuron

$$a = \sum_{i=1}^D w_i x_i + w_0$$

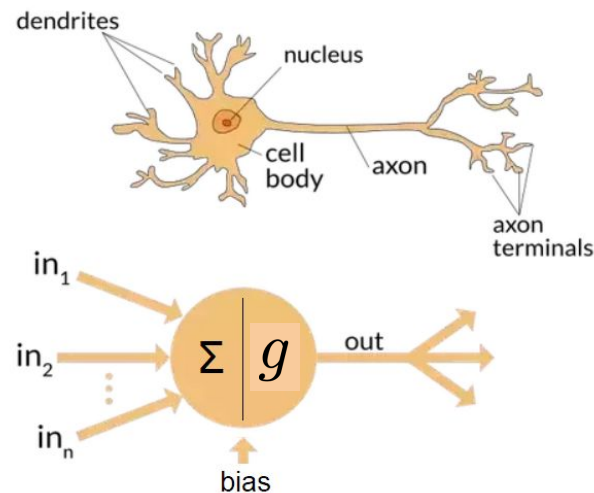


Image from
<https://towardsdatascience.com/the-differences-between-artificial-and-biological-neural-networks-a8b46db828b7>

Computation within a single neuron

- Activation function

$$h = g(a)$$

- ReLU activation function:

$$g(a) = \max(a, 0)$$

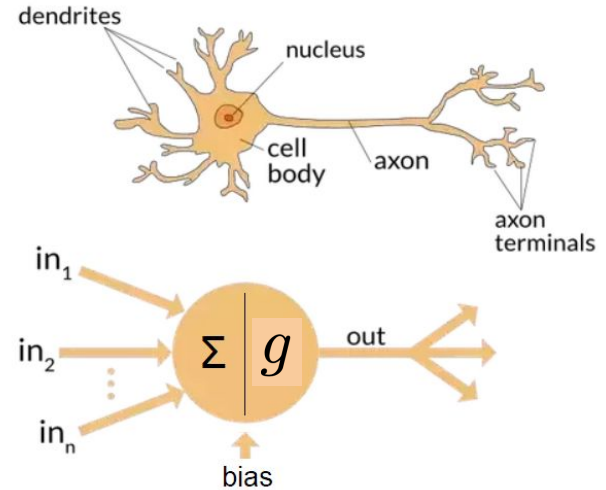
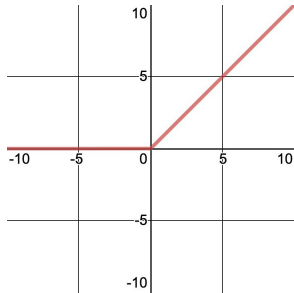


Image from
<https://towardsdatascience.com/the-differences-between-artificial-and-biological-neural-networks-a8b46db828b7>

Computation within a single neuron

One neuron:

$$a = \sum_{i=1}^D w_i x_i + w_0$$

$$a = \mathbf{w}^\top \mathbf{x}$$

$$\mathbf{x} = [x_1, \dots, x_D, 1]$$

Single layer with multiple neurons

Let's add more neurons:

$$a_{\mathbf{k}} = \sum_{i=1}^D w_{\mathbf{k}i} x_i + w_{\mathbf{k}0}$$

$$a_{\mathbf{k}} = \mathbf{w}_{\mathbf{k}}^{\top} \mathbf{x}$$

$$\mathbf{x} = [x_1, \dots, x_D, 1]$$

Single layer with multiple neurons

Rewrite as matrix multiplication and add ReLU non-linearity:

$$\mathbf{a} = W\mathbf{x}, \quad W \in \mathbb{R}^{200 \times 785}$$

$$h_i = \max(a_i, 0), \quad i = 1, \dots, 200$$

```
model = tf.keras.Sequential([
    tf.keras.layers.Flatten(input_shape=(28, 28, 1)),
    tf.keras.layers.Dense(200, activation='relu'),
    tf.keras.layers.Dense(60, activation='relu'),
    tf.keras.layers.Dense(10, activation='softmax') # class
])

# this configures the training of the model. Keras calls it
model.compile(optimizer='adam',
              loss='categorical_crossentropy',
              metrics=['accuracy'])

# train the model
model.fit(training_dataset, ...)
```

Multiple layers

Feed output as input to the next layer:

$$\mathbf{a}^{(k)} = W^{(k)} \mathbf{h}_{k-1}, \quad W \in \mathbb{R}^{N_k \times N_{k-1}}$$

$$h_i^{(k)} = \max(a_i^{(k)}, 0), \quad i = 1, \dots, N_k$$

```
model = tf.keras.Sequential([
    tf.keras.layers.Flatten(input_shape=(28, 28, 1)),
    tf.keras.layers.Dense(200, activation='relu'),
    tf.keras.layers.Dense(60, activation='relu'),
    tf.keras.layers.Dense(10, activation='softmax') # class
])

# this configures the training of the model. Keras calls it
model.compile(optimizer='adam',
               loss='categorical_crossentropy',
               metrics=['accuracy'])

# train the model
model.fit(training_dataset, ...)
```

Final layer with softmax activation

Last layer should generate a 10-dimensional vector with probability of each digit

$$\mathbf{a}^{(k)} = W^{(k)} \mathbf{h}_{k-1}, \quad W \in \mathbb{R}^{10 \times N_{k-1}}$$

$$\hat{y}_i = \frac{\exp(a_i^{(k)})}{\sum_j \exp(a_j^{(k)})}, \quad i = 1, \dots, 10$$

```
model = tf.keras.Sequential([
    tf.keras.layers.Flatten(input_shape=(28, 28, 1)),
    tf.keras.layers.Dense(200, activation='relu'),
    tf.keras.layers.Dense(60, activation='relu'),
    tf.keras.layers.Dense(10, activation='softmax') # class probabilities
])

# this configures the training of the model. Keras calls it
model.compile(optimizer='adam',
              loss='categorical_crossentropy',
              metrics=['accuracy'])

# train the model
model.fit(training_dataset, ...)
```

Final layer with softmax activation

Last layer should generate a 10-dimensional vector with probability of each digit

$$\mathbf{a}^{(k)} = W^{(k)} \mathbf{h}_{k-1}, \quad W \in \mathbb{R}^{10 \times N_{k-1}}$$

$$\hat{y}_i = \text{softmax}(\mathbf{a}^{(k)})_i, \quad i = 1, \dots, 10$$

```
model = tf.keras.Sequential([
    tf.keras.layers.Flatten(input_shape=(28, 28, 1)),
    tf.keras.layers.Dense(200, activation='relu'),
    tf.keras.layers.Dense(60, activation='relu'),
    tf.keras.layers.Dense(10, activation='softmax') # class probabilities
])

# this configures the training of the model. Keras calls it
model.compile(optimizer='adam',
               loss='categorical_crossentropy',
               metrics=['accuracy'])

# train the model
model.fit(training_dataset, ...)
```

Final layer with softmax activation

Last layer should generate a 10-dimensional vector with probability of each digit

$$\mathbf{z} = W^{(k)} \mathbf{h}_{k-1}, \quad W \in \mathbb{R}^{10 \times N_{k-1}}$$

$$\hat{y}_i = \text{softmax}(\mathbf{z})_i, \quad i = 1, \dots, 10$$

```
model = tf.keras.Sequential([
    tf.keras.layers.Flatten(input_shape=(28, 28, 1)),
    tf.keras.layers.Dense(200, activation='relu'),
    tf.keras.layers.Dense(60, activation='relu'),
    tf.keras.layers.Dense(10, activation='softmax') # class probabilities
])

# this configures the training of the model. Keras calls it
model.compile(optimizer='adam',
              loss='categorical_crossentropy',
              metrics=['accuracy'])

# train the model
model.fit(training_dataset, ...)
```

Categorical cross-entropy loss

- Training example:

$$(\mathbf{x}, y) \quad y \in 1, \dots, 10$$

- Categorical cross-entropy loss:

$$\begin{aligned} \log p(y|\mathbf{x}; \theta) &= \log \text{softmax}(\mathbf{z})_y \\ &= \mathbf{z}_y - \log \sum_{j=1}^{10} \exp(\mathbf{z}_j) \end{aligned}$$

```
model = tf.keras.Sequential([
    tf.keras.layers.Flatten(input_shape=(28, 28, 1)),
    tf.keras.layers.Dense(200, activation='relu'),
    tf.keras.layers.Dense(60, activation='relu'),
    tf.keras.layers.Dense(10, activation='softmax') # class
])

# this configures the training of the model. Keras calls it
model.compile(optimizer='adam',
              loss='categorical_crossentropy',
              metrics=['accuracy'])

# train the model
model.fit(training_dataset, ...)
```


Model training

- Training example:

$$(\mathbf{x}, y) \quad y \in 1, \dots, 10$$

- Categorical cross-entropy loss:

$$\begin{aligned} \log p(y|\mathbf{x}; \theta) &= \log \text{softmax}(\mathbf{z})_y \\ &= \mathbf{z}_y - \log \sum_{j=1}^{10} \exp(\mathbf{z}_j) \end{aligned}$$

- For a training dataset $\mathcal{D} = \{(\mathbf{x}_d, y_d)\}$, $d = 1, \dots, N$

$$\mathcal{L}(\theta) = \frac{1}{N} \sum_d \log(y_d|\mathbf{x}_d; \theta)$$

```
model = tf.keras.Sequential([
    tf.keras.layers.Flatten(input_shape=(28, 28, 1)),
    tf.keras.layers.Dense(200, activation='relu'),
    tf.keras.layers.Dense(60, activation='relu'),
    tf.keras.layers.Dense(10, activation='softmax') # class
])

# this configures the training of the model. Keras calls it
model.compile(optimizer='adam',
              loss='categorical_crossentropy',
              metrics=['accuracy'])

# train the model
model.fit(training_dataset, ...)
```

Take a quiz!