Sapienza Training Camp 2020

Building an Image Search Engine

3 - 5 September, 2020

Roadmap

Image from the photo collection

I_k

Neural Network (CNN)

Image encoding

φ_k

Recurrent neural network (RNN)

Image description

T_k = “dog next to person on the bike near street crossing”

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

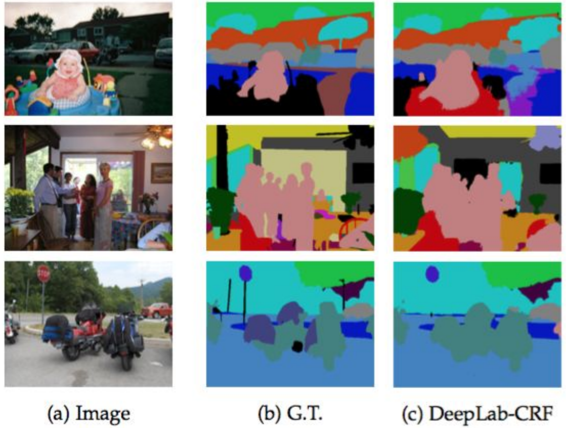
Q Query: “person walking with a dog on the beach”

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

Agenda for today

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

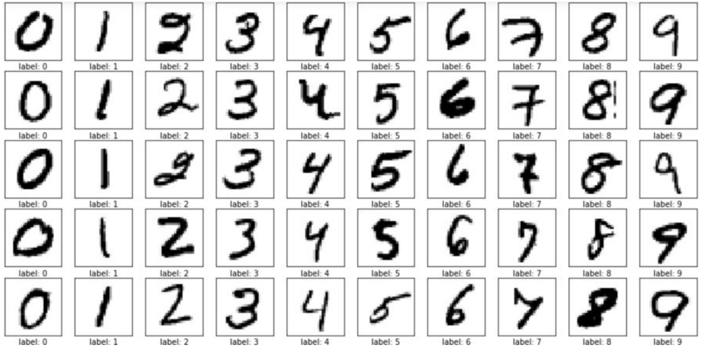
Computer Vision



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

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

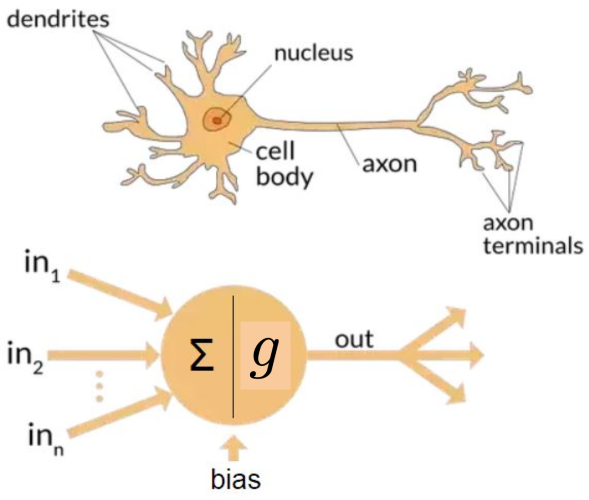


Image from https://towardsdatascience.com/the-differences-between-artificial-and-biological-neural-networks-a8b46db828b7

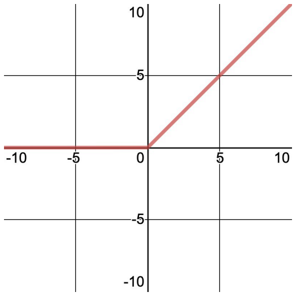
Computation within a single neuron

- Activation function

$$h = g(a)$$

- ReLU activation function:

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



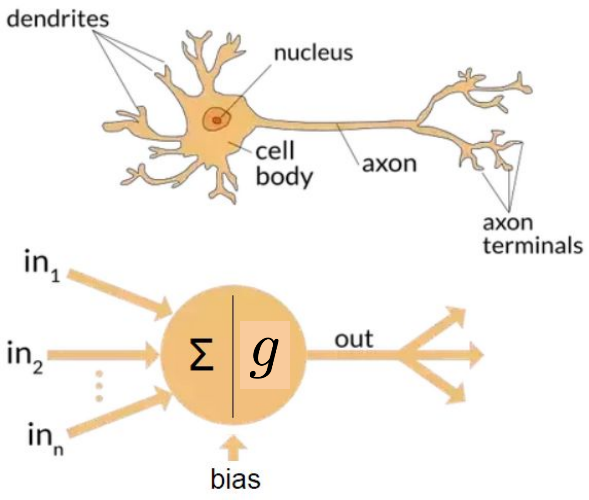


Image from https://towardsdatascience.com/the-differences-between-artificial-and-biological-neural-networks-a8b46db828b7

Computation within a single neuron

One neuron:

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

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

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

Single layer with multiple neurons

Let’s add more neurons:

$$a_k = \sum_{i=1}^{D} w_{ki} x_i + w_{k0}$$

$$a_k = {\mathbf{w}_k}^\top \mathbf{x}$$

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

Single layer with multiple neurons

Rewrite as matrix multiplication and add ReLU non-linearity:

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

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

```
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') # cla
  ])

# 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:

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

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

```
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') # cla
  ])

# 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

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

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

```
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') # cla
  ])

# 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

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

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

```
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') # cla
  ])

# 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

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

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

```
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') # cla
  ])

# 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:

$$(\mathbf{x}, y) \qquad y \in 1, \ldots, 10$$

- Categorical cross-entropy loss:

$$\log p(y|\mathbf{x};\theta) = \log \mathrm{softmax}(\mathbf{z})_y$$

$$= \mathbf{z}_y - \log \sum_{j=1}^{10} \exp(\mathbf{z}_j)$$

```
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') # cla
  ])

# 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:

  $(\mathbf{x}, y) \qquad y \in 1, \ldots, 10$

- Categorical cross-entropy loss:

$$\log p(y|\mathbf{x};\theta) = \log \operatorname{softmax}(\mathbf{z})_y$$

$$= \mathbf{z}_y - \log \sum_{j=1}^{10} \exp(\mathbf{z}_j)$$

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

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

```
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') # cla
  ])

# 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!