Lab 12-2: Image Captioning

Department of Computer Science,
National Tsing Hua University, Taiwan
2022

Outline

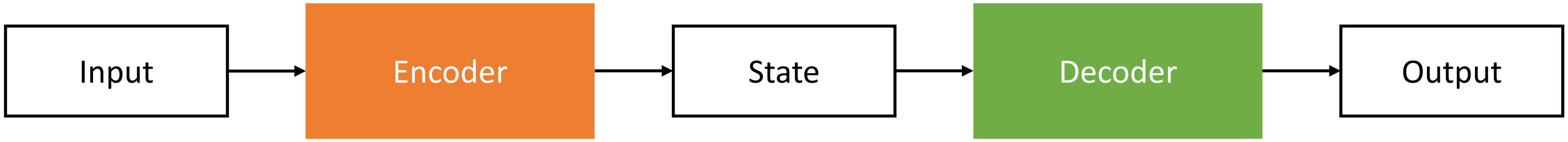
- Encoder-Decoder model
- Attention-based
- Assignment

Outline

- Encoder-Decoder model
- Attention-based
- Assignment

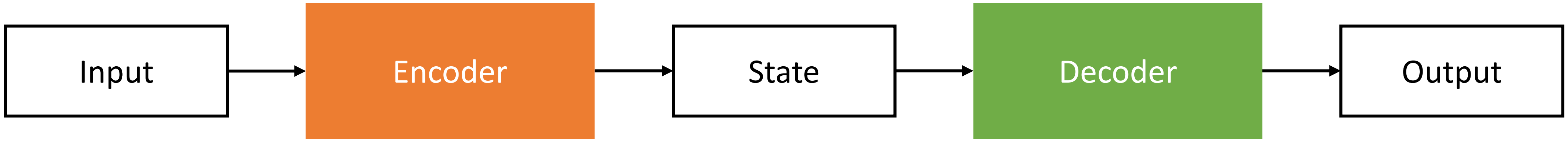
Encoder-Decoder Model

- Lab12-1: Neural Machine Translation
  - Encoder RNN: reads the source sentence and transforms it into a rich fixed-length vector representation
  - Decoder RNN: uses the representation as the initial hidden state and generates the target sentence



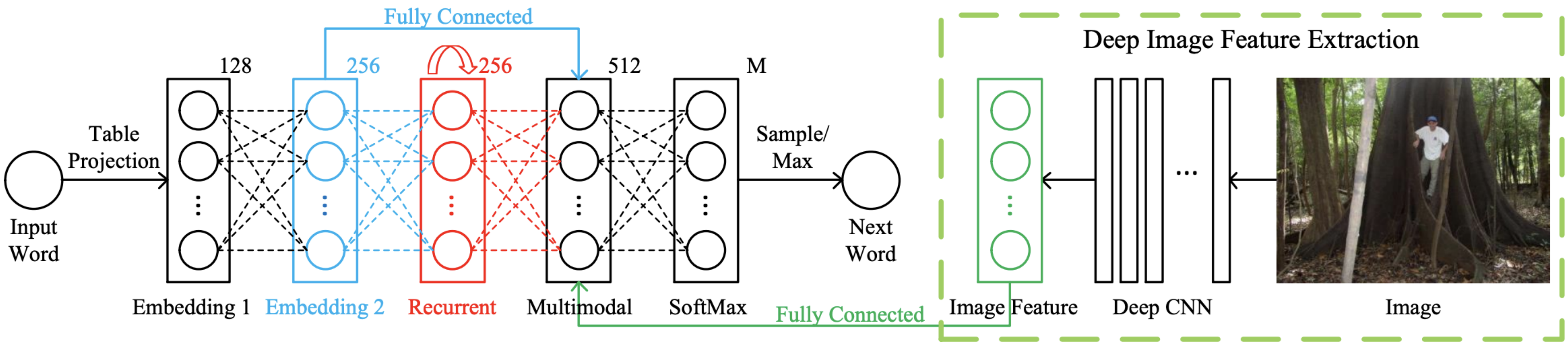
Encoder-Decoder Model

- Image Captioning
  - Encoder CNN: reads the images and transforms it into a rich fixed-length vector representation
  - Decoder RNN: uses the representation as the initial hidden state and generates the target sentence



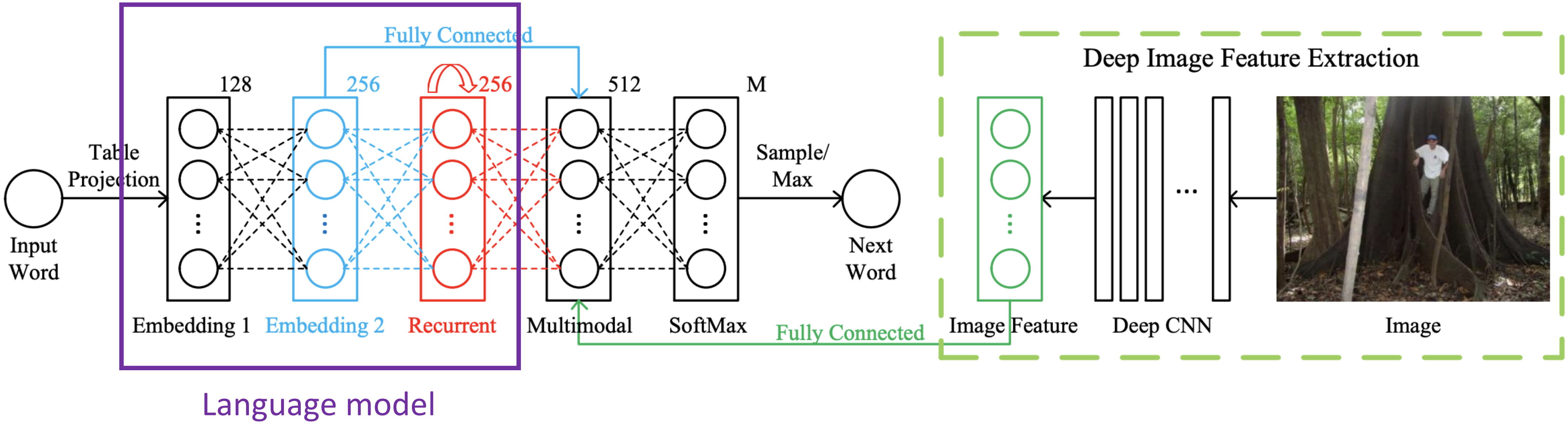
Encoder-Decoder Model

- m-RNN (multimodal RNN)



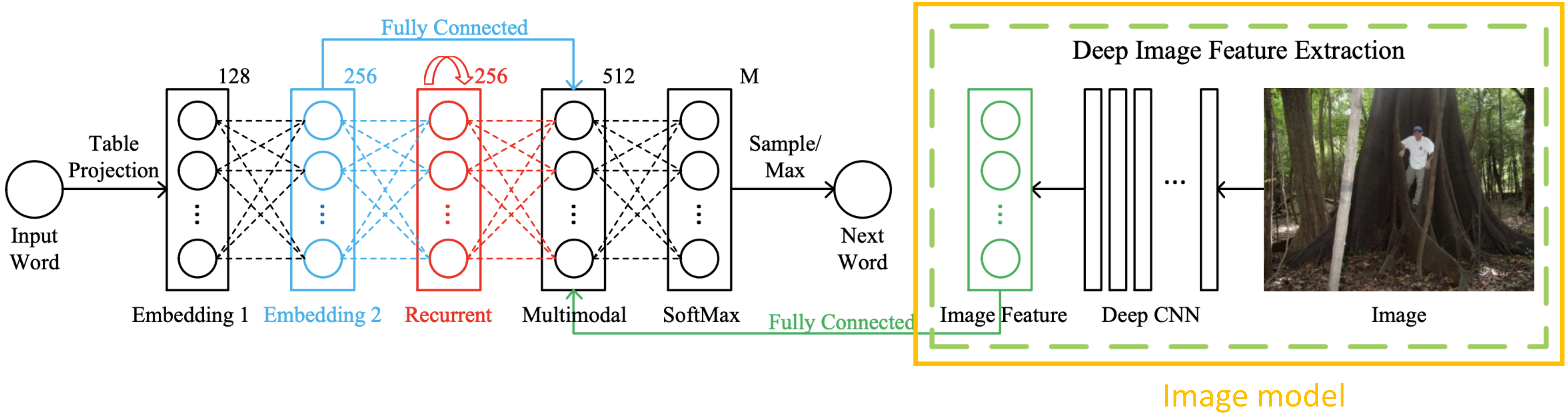
Encoder-Decoder Model

- m-RNN (multimodal RNN)
  - The language model part learns the dense feature embedding for each word



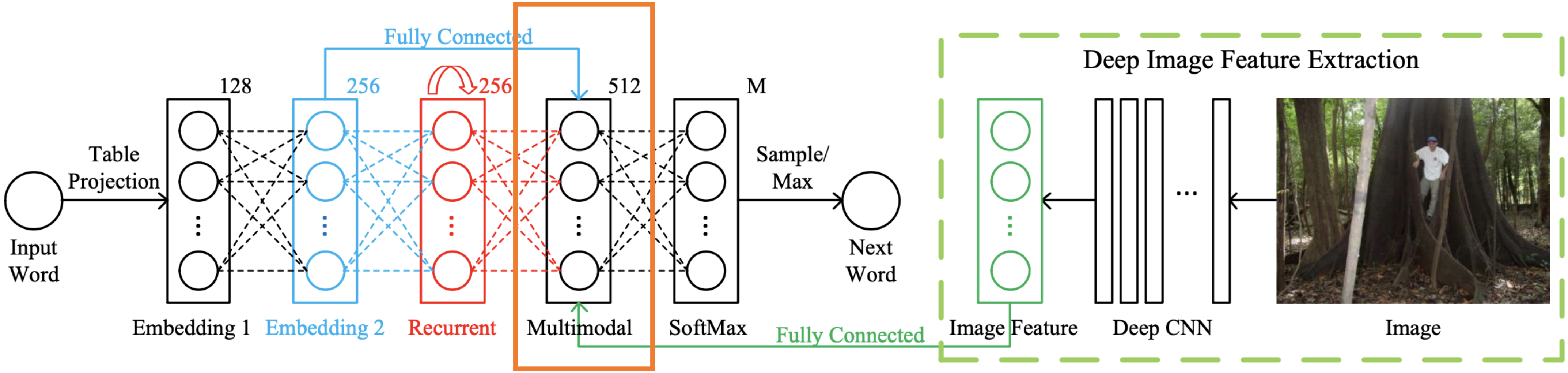
Encoder-Decoder Model

- m-RNN (multimodal RNN)
  - The language model part learns the dense feature embedding for each word
  - The image part contains a deep CNN which extracts image features



Encoder-Decoder Model

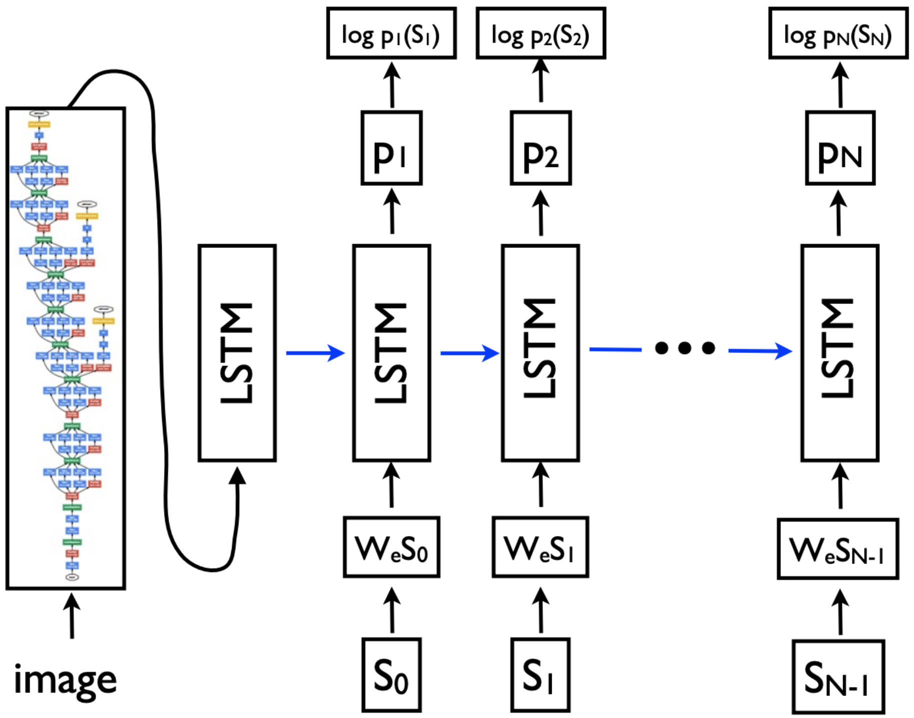
- m-RNN (multimodal RNN)
  - The language model part learns the dense feature embedding for each word
  - The image part contains a deep CNN which extracts image features
  - The multimodal part connects the language model and the deep CNN together by a one-layer representation



Multimodal model

Encoder-Decoder Model

- NIC
  - A generative model based on a deep recurrent architecture that combines recent advances in computer vision and machine translation
  - Uses a more powerful CNN in the encoder
  - The image is only input once

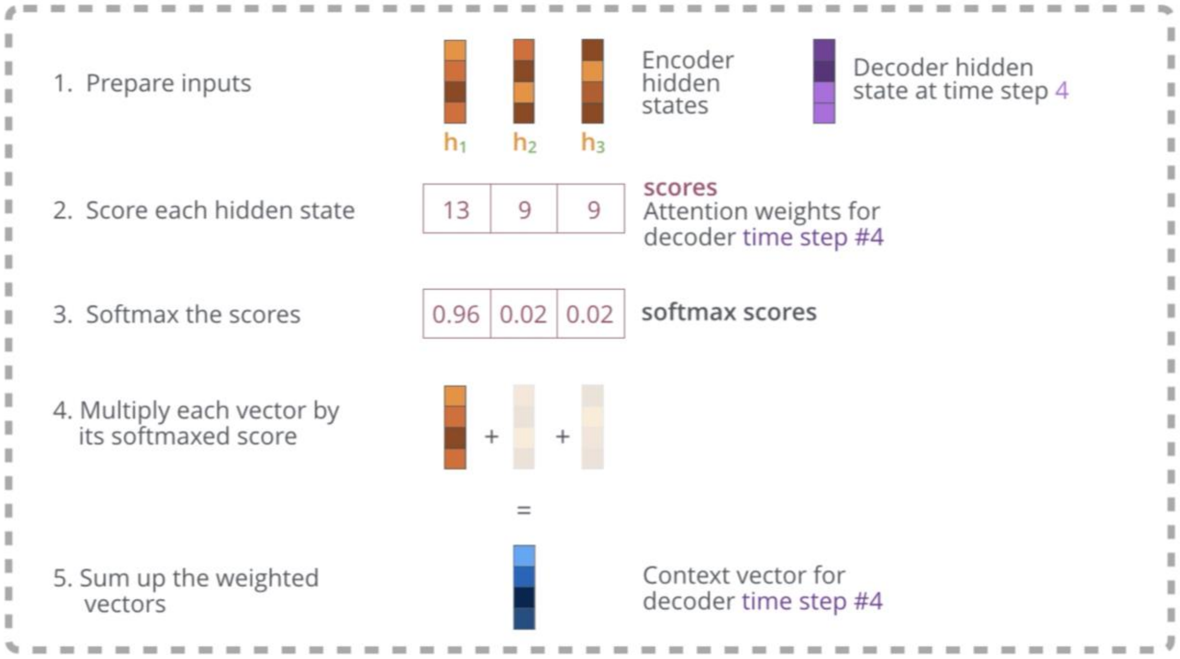


Outline

- Encoder-Decoder model
- Attention-based
- Assignment

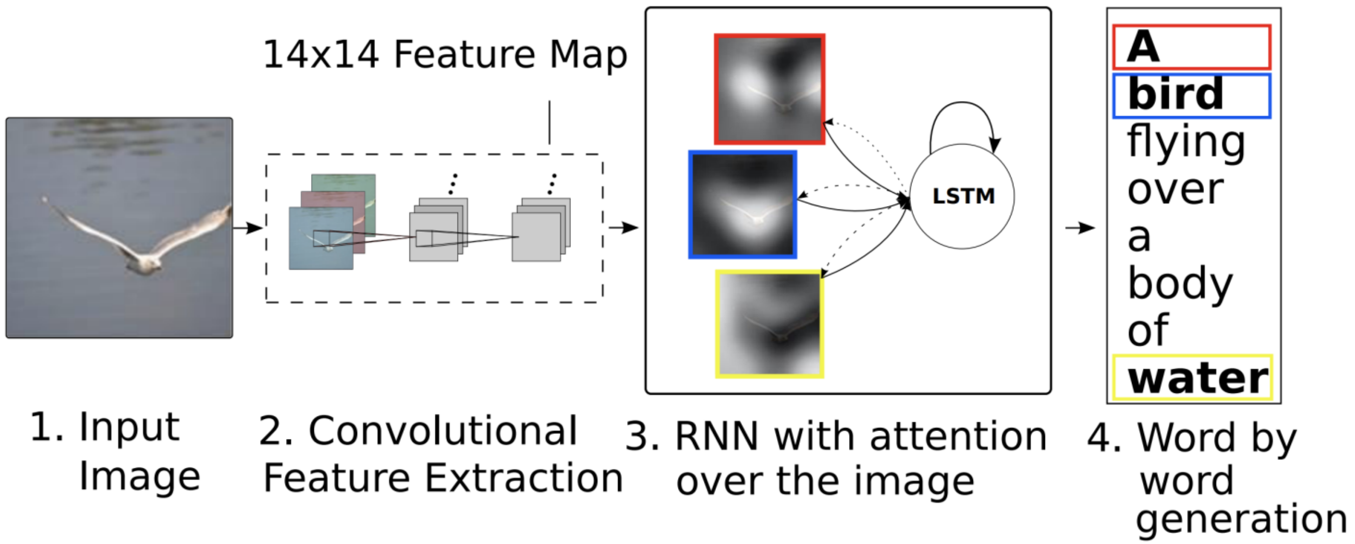
Attention Based

- Attention allows the model to focus on the relevant parts of the input sequence as needed



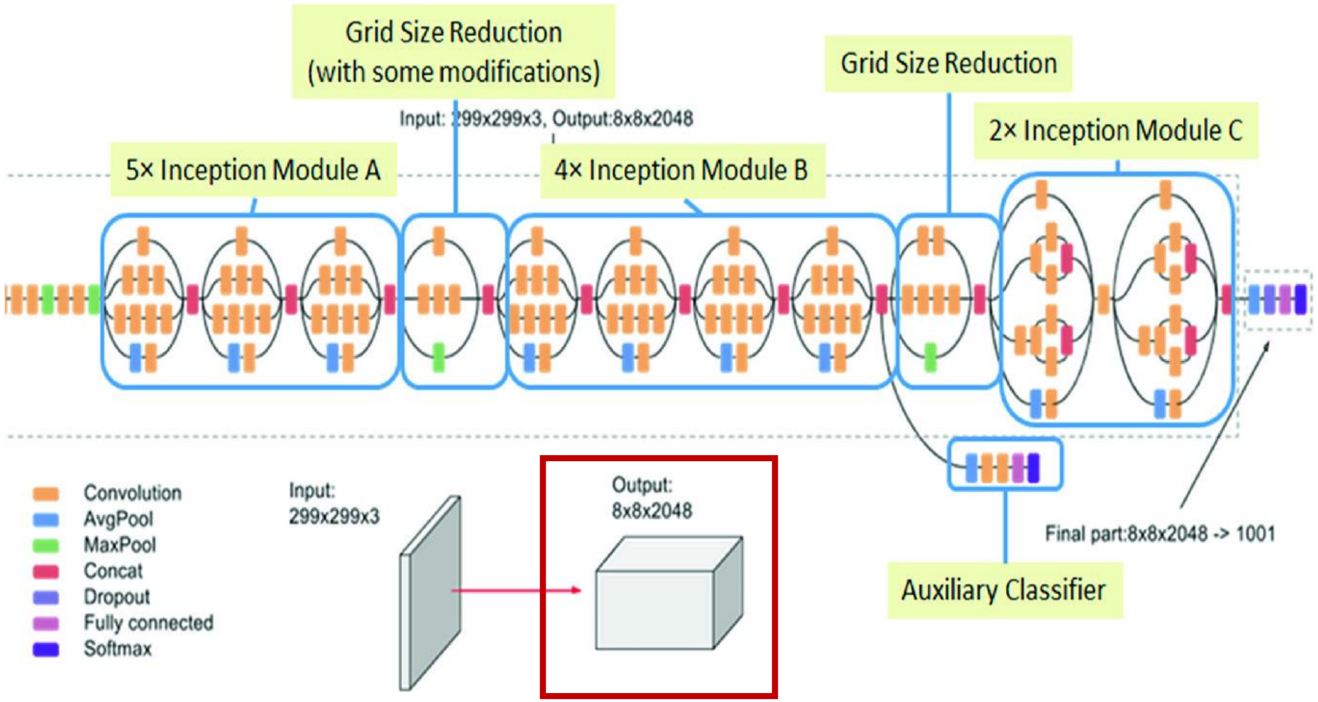
Attention Based

- Attention allows the model to focus on the relevant parts of the input sequence as needed
  - Show, Attend and Tell: Neural Image Caption Generation with Visual Attention



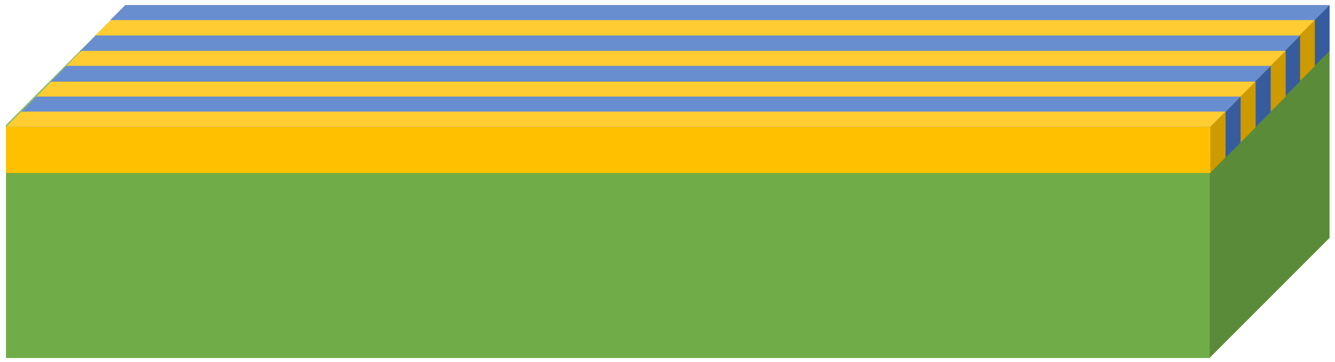
Attention Based

- First, extract the features from image by Inception-v3



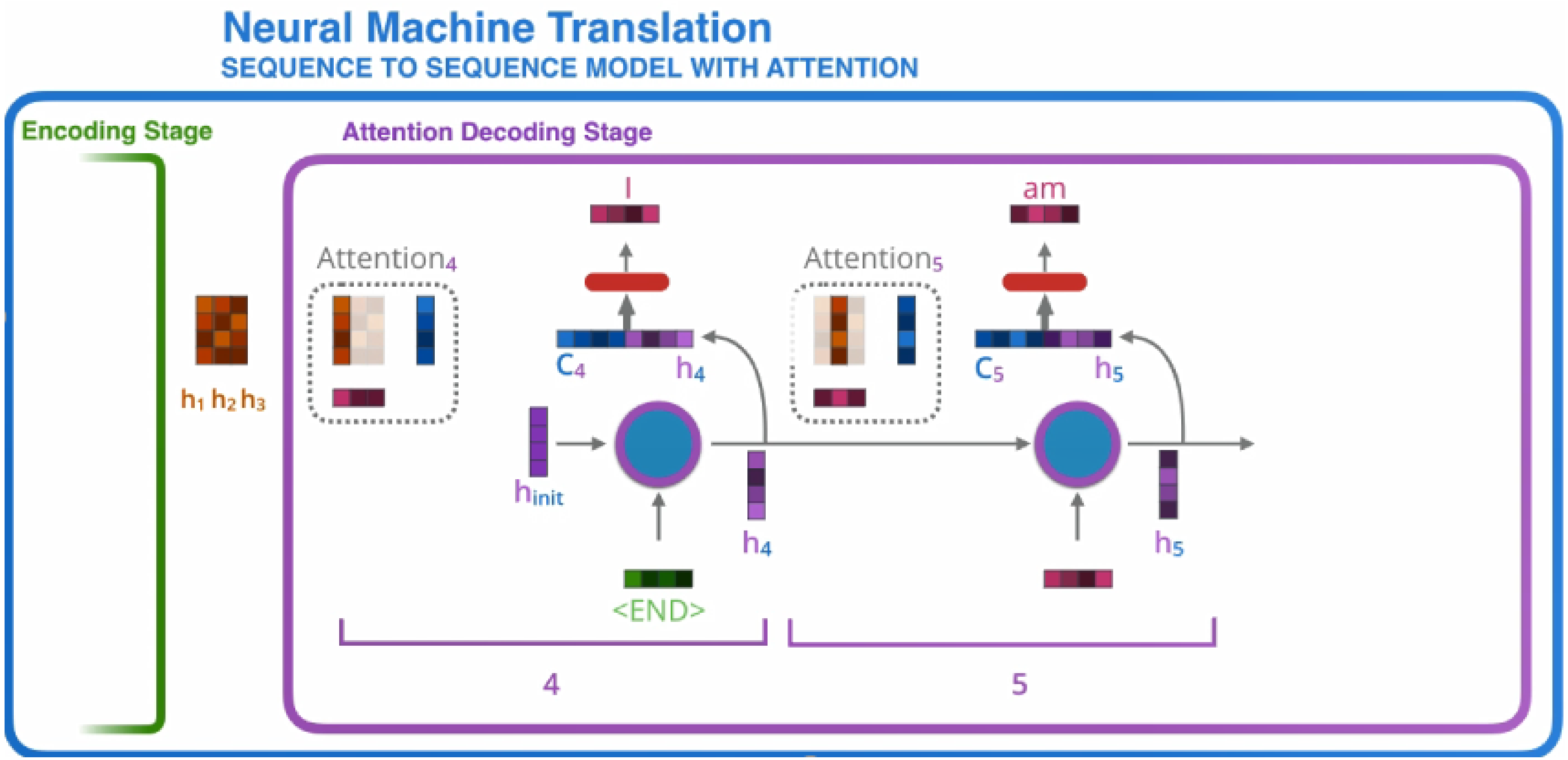
Attention Based

- First, extract the features from image by Inception-v3
- We have a 8*8*2048 size feature map, the last layer has 8*8 pixel locations which corresponds to certain portion in image
- That means we have 64 pixel locations
- The model will then learn an attention over these locations



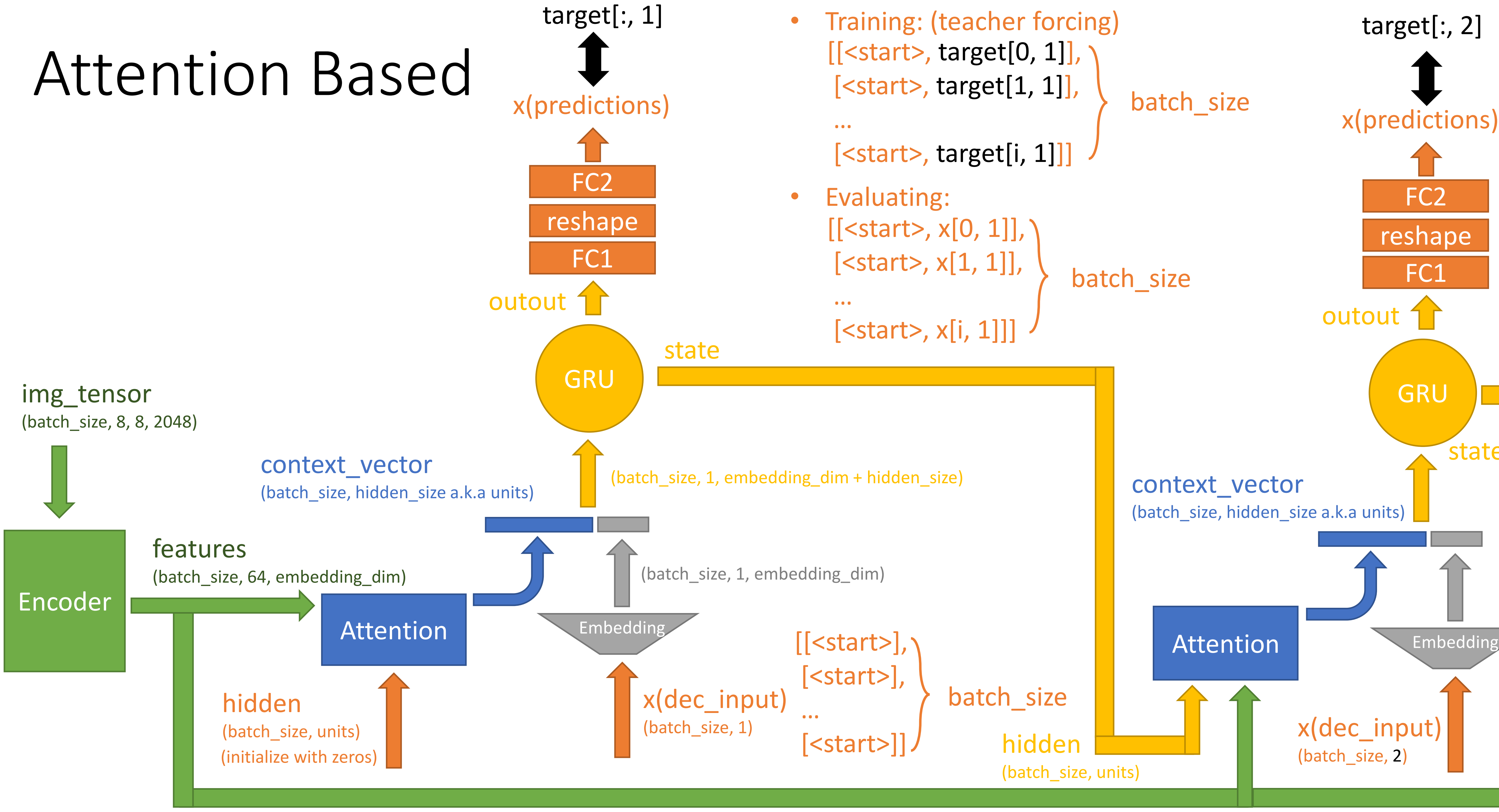
Attention Based

- The rest is similar to the neural machine translation task



Animation: https://jalammar.github.io/visualizing-neural-machine-translation-mechanics-of-seq2seq-models-with-attention

Attention Based



- Training: (teacher forcing)
  [[<start>, target[0, 1]],
  ...
  [<start>]]

target[:, 2]

```python
class RNN_Decoder(tf.keras.Model):
    def __init__(self, embedding_dim, units, vocab_size):
        super(RNN_Decoder, self).__init__()
        self.units = units

        self.embedding = tf.keras.layers.Embedding(vocab_size, embedding_dim)
        self.gru = tf.keras.layers.GRU(self.units,
                                       return_sequences=True,
                                       return_state=True,
                                       recurrent_initializer='glorot_uniform')
        self.fc1 = tf.keras.layers.Dense(self.units)
        self.fc2 = tf.keras.layers.Dense(vocab_size)

        self.attention = BahdanauAttention(self.units)

    def call(self, x, features, hidden):
        # defining attention as a separate model
        context_vector, attention_weights = self.attention(features, hidden)

        # x shape after passing through embedding == (batch_size, 1, embedding_dim)
        x = self.embedding(x)

        # x shape after concatenation == (batch_size, 1, embedding_dim + hidden_size)
        x = tf.concat([tf.expand_dims(context_vector, 1), x], axis=-1)

        # passing the concatenated vector to the GRU
        output, state = self.gru(x)

        # shape == (batch_size, max_length, hidden_size)
        x = self.fc1(output)

        # x shape == (batch_size * max_length, hidden_size)
        x = tf.reshape(x, (-1, x.shape[2]))

        # output shape == (batch_size * max_length, vocab)
        x = self.fc2(x)

        return x, state, attention_weights
```

```python
@tf.function
def train_step(img_tensor, target):
    loss = 0

    # initializing the hidden state for each batch
    # because the captions are not related from image to image
    hidden = decoder.reset_state(batch_size=target.shape[0])

    dec_input = tf.expand_dims([tokenizer.word_index['<start>']] * BATCH_SIZE, 1)

    with tf.GradientTape() as tape:
        features = encoder(img_tensor)

        for i in range(1, target.shape[1]):
            # passing the features through the decoder
            predictions, hidden, _ = decoder(dec_input, features, hidden)

            loss += loss_function(target[:, i], predictions)

            # using teacher forcing
            dec_input = tf.expand_dims(target[:, i], 1)

    total_loss = (loss / int(target.shape[1]))

    trainable_variables = encoder.trainable_variables + decoder.trainable_variables

    gradients = tape.gradient(loss, trainable_variables)

    optimizer.apply_gradients(zip(gradients, trainable_variables))

    return loss, total_loss
```

img

Enc

hidden
(batch_size, units)

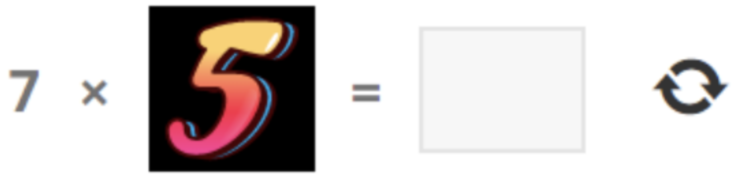
x(dec_input)
(batch_size, 2)

Outline

- Encoder-Decoder model
- Attention-based
- Assignment

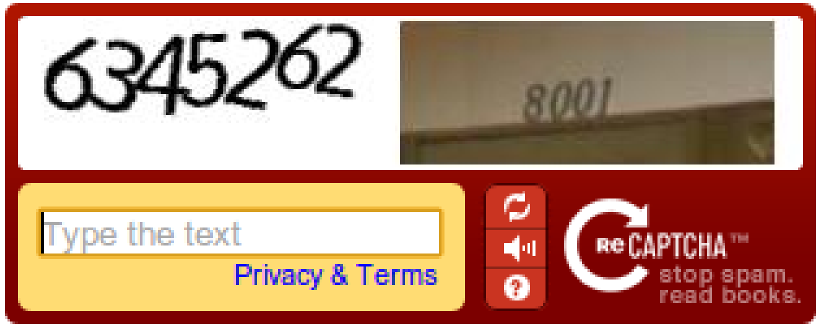
Assignment

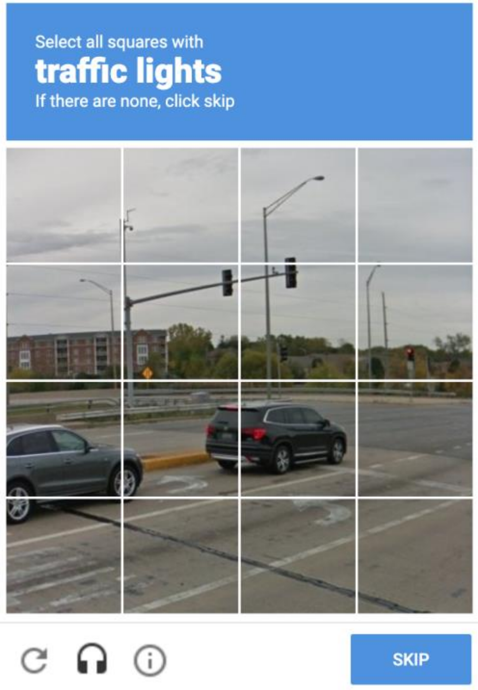
- CAPTCHA
  - An acronym for “Completely Automated Public Turing test to tell Computers and Humans Apart”
  - A type of challenge–response test used in computing to determine whether or not the user is human
  - Prevents spam attacks and protects websites from bots



Assignment

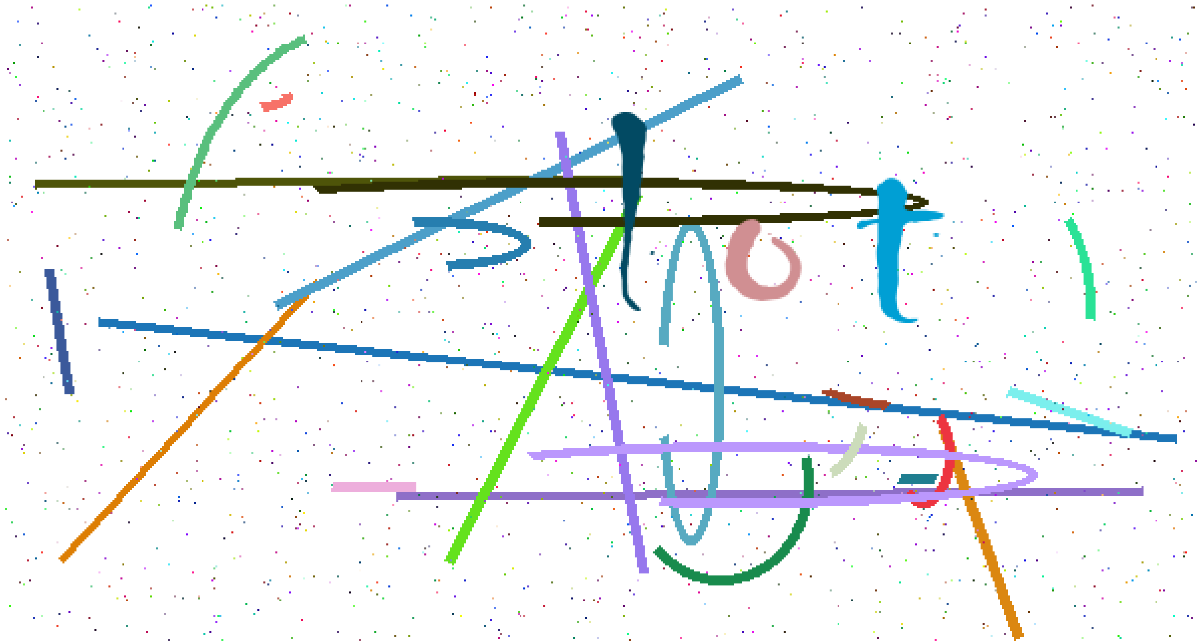
- reCAPTCHA
  - Establish that a computer user is human
  - Assist in the digitization of books or improve machine learning





Assignment

- We are going to train a captcha recognizer in this lab
- Dataset
  - 140,000 CAPTCHAs



Assignment

- Requirement
  - Use any model architectures you want
  - Design your own model architecture
  - The first 100,000 as training data, the next 20,000 as validation data, and the rest as testing data
  - Only if the whole word matches exactly does it count as correct
  - Predict the answer to the testing data and write them in a file
  - Testing accuracy should be at least 90%
- Please submit your code file and the answer file

Assignment

- Requirement
  - Use any model architectures you want
  - Design your own model architecture
  - The first 100,000 as training data, the next 20,000 as validation d
    rest as testing data
  - Only if the whole word matches exactly does it count as correct
  - Predict the answer to the testing data and write them in a file
  - Testing accuracy should be at least 90%
- Please submit your code file and the answer file

```
a0 thus
a1 www
a2 tied
a3 ids
a4 jam
a5 zoo
a6 apple
a7 big
a8 lot
a9 above
a10 ooo
```