# CSCI E-89B Introduction to Natural Language Processing

Harvard Extension School

Dmitry Kurochkin

Fall 2024
Lecture 4

# Contents

1. Bag of Words (BoW)
   - BoW Representation
   - Hands-On: BoW in Python
     - Sklearn
     - NLTK
     - SpaCy
   - Limitations of BoW
2. Introduction to n-grams
   - n-grams Overview
   - Creating n-grams in Python: Sklearn, NLTK, and SpaCy
   - Limitations of n-grams
3. Convolutional Neural Networks (CNN)
   - Convolution
   - CNN Layers
     - Convolutional Layers
     - MaxPooling Layer
   - Building CNN in Python

# Contents

1. Bag of Words (BoW)
   - BoW Representation
   - Hands-On: BoW in Python
     - Sklearn
     - NLTK
     - SpaCy
   - Limitations of BoW
2. Introduction to n-grams
   - n-grams Overview
   - Creating n-grams in Python: Sklearn, NLTK, and SpaCy
   - Limitations of n-grams
3. Convolutional Neural Networks (CNN)
   - Convolution
   - CNN Layers
     - Convolutional Layers
     - MaxPooling Layer
   - Building CNN in Python

# Bag of Words (BoW) Representation

- Bag of Words (BoW)
  - Bag of Words (BoW) is a technique to represent text data as numerical vectors.
  - It serves as a basic and yet powerful approach for text representation.
  - Each document is considered as a collection of words, disregarding grammar and word order.

- Application of BoW
  - Facilitates text classification and clustering.
    - ★ Creates a feature space for documents used in various machine learning algorithms.
  - Supports sentiment analysis and topic modeling.
    - ★ Useful for analyzing document sentiment and classifying documents into topics.
  - Simplifies text data into quantifiable features.
    - ★ Converts qualitative text data into quantitative vectors for use in ML models.

# Steps in Creating a BoW Representation

1. **Tokenization:**
   - Divide the text into individual units (tokens): phrases, words, subwords, or characters
   - Example:
     "Henry Ford introduced the Model T. Ford Model T was revolutionary."
     → ['Henry', 'Ford', 'introduced', 'the', 'Model', 'T', 'was', 'revolutionary']
2. **Vocabulary Creation:**
   - Build a vocabulary of unique words from the entire text corpus
   - Example:
     {"Henry", "Ford", "introduced", "the", "Model", "T", "was", "revolutionary"}
3. **Frequency Counting:**
   - For each document, count the frequency of each word from the vocabulary
   - Example:
     "Henry Ford introduced the Model T."
     →{"Henry": 1, "Ford": 2, "introduced": 1, "the": 1, "Model": 2, "T": 2, "was": 1, "revolutionary": 1}

# Contents

1. Bag of Words (BoW)
    - BoW Representation
    - Hands-On: BoW in Python
        - Sklearn
        - NLTK
        - SpaCy
    - Limitations of BoW
2. Introduction to n-grams
    - n-grams Overview
    - Creating n-grams in Python: Sklearn, NLTK, and SpaCy
    - Limitations of n-grams
3. Convolutional Neural Networks (CNN)
    - Convolution
    - CNN Layers
        - Convolutional Layers
        - MaxPooling Layer
    - Building CNN in Python

# Hands-On: BoW in Python Using Sklearn

```python
from sklearn.feature_extraction.text import CountVectorizer

# Example text
corpus = ["Henry Ford introduced the Model T. Ford Model T was revolutionary."]

# Initialize CountVectorizer
vectorizer = CountVectorizer()

# Fit and Transform the corpus
X = vectorizer.fit_transform(corpus)

# Display the Vocabulary
print("Vocabulary:\n", vectorizer.get_feature_names_out(),"\n")

# Display the BoW Representation
print("BoW Representation:", X.toarray())
```

```
Vocabulary:
['ford' 'henry' 'introduced' 'model' 'revolutionary' 'the' 'was']

BoW Representation: [[2 1 1 2 1 1 1]]
```

# Hands-On: BoW in Python Using NLTK

```python
import nltk
from nltk.tokenize import word_tokenize
nltk.download('punkt')  # This will download the required tokenizer models

# Example text
text = "Henry Ford introduced the Model T. Ford Model T was revolutionary."

# Tokenize the text
tokens = word_tokenize(text)

# Create vocabulary (unique tokens)
vocabulary = list(set(tokens))

# Create Bag of Words representation
bow_vector = [tokens.count(word) for word in vocabulary]

# Display results
print("Vocabulary:\n", vocabulary)
print("BoW Representation:", bow_vector, "\n")
```

```
Vocabulary:
['T', 'was', 'Ford', 'introduced', 'Henry', 'T.', '.', 'the',
'revolutionary', 'Model']

BoW Representation: [1, 1, 2, 1, 1, 1, 1, 1, 1, 2]
```

# Hands-On: BoW in Python Using SpaCy

```
import spacy

# Load SpaCy model
nlp = spacy.load("en_core_web_sm")

# Example text
text = "Henry Ford introduced the Model T. Ford Model T was revolutionary."

# Process the text with SpaCy to create a Doc object
doc = nlp(text)

# Tokenize the text
tokens = [token.text for token in doc]

# Create vocabulary (unique tokens)
vocabulary = list(set(tokens))

# Create Bag of Words representation
bow_vector = [tokens.count(word) for word in vocabulary]

# Display results
print("Vocabulary:", vocabulary, "\n")
print("BoW Representation:", bow_vector)
```

```
Vocabulary: [’T’, ’was’, ’Ford’, ’introduced’, ’Henry’,
’T.’, ’.’, ’the’, ’revolutionary’, ’Model’]

BoW Representation: [1, 1, 2, 1, 1, 1, 1, 1, 1, 2]
```

# Contents

1. Bag of Words (BoW)
    - BoW Representation
    - Hands-On: BoW in Python
        - Sklearn
        - NLTK
        - SpaCy
    - Limitations of BoW
2. Introduction to n-grams
    - n-grams Overview
    - Creating n-grams in Python: Sklearn, NLTK, and SpaCy
    - Limitations of n-grams
3. Convolutional Neural Networks (CNN)
    - Convolution
    - CNN Layers
        - Convolutional Layers
        - MaxPooling Layer
    - Building CNN in Python

# Limitations of Bag of Words (BoW)

- Limitations
  - Ignores word order and context.
  - Creates sparse representations with high-dimensional vectors.
  - Treats words as independent features, losing semantic meaning.
  - Inefficient for large vocabularies due to high storage and computational requirements.

- Alternatives
  - Use embeddings or context-aware models.
    - ★ Example: Word2Vec, GloVe, BERT, Transformers.
    - ★ Capture semantic meaning and context.
  - Use Recurrent Neural Networks (RNNs).
    - ★ Capture sequential dependencies and context over time.
    - ★ Suitable for tasks like language modeling, text generation, and sentiment analysis.
  - Use Convolutional Neural Networks (CNNs).
    - ★ Capture local patterns and features in text sequences.
    - ★ Effective for text classification, sentiment analysis, and other tasks.

# Contents

1. Bag of Words (BoW)
   - BoW Representation
   - Hands-On: BoW in Python
     - Sklearn
     - NLTK
     - SpaCy
   - Limitations of BoW
2. Introduction to n-grams
   - n-grams Overview
   - Creating n-grams in Python: Sklearn, NLTK, and SpaCy
   - Limitations of n-grams
3. Convolutional Neural Networks (CNN)
   - Convolution
   - CNN Layers
     - Convolutional Layers
     - MaxPooling Layer
   - Building CNN in Python

# n-grams Overview

- n-grams
  - Consecutive sequences of n items from a text.
  - Types include unigrams (1-gram), bigrams (2-gram), trigrams (3-gram), and higher n-grams.
  - Capture adjacent word relationships.
- Importance of n-grams in NLP
  - Capture local context around words.
  - Aid in creating language models.
  - Enhance text classification and generation algorithms.
- Practical Use Cases
  - Text prediction in keyboards and search engines.
  - Spell checkers and grammar correctors.
  - Machine translation systems: Translate sequences of words rather than isolated words.
  - Sentiment analysis: Capture sentiment-carrying phrases for better analysis.

# Contents

1. Bag of Words (BoW)
   - BoW Representation
   - Hands-On: BoW in Python
     - Sklearn
     - NLTK
     - SpaCy
   - Limitations of BoW
2. Introduction to n-grams
   - n-grams Overview
   - Creating n-grams in Python: Sklearn, NLTK, and SpaCy
   - Limitations of n-grams
3. Convolutional Neural Networks (CNN)
   - Convolution
   - CNN Layers
     - Convolutional Layers
     - MaxPooling Layer
   - Building CNN in Python

# Creating 2-grams in Python: Sklearn

```
from sklearn.feature_extraction.text import CountVectorizer

# Example text
corpus = ["Henry Ford introduced the Model T. Ford Model T was revolutionary."]

# Initialize CountVectorizer with ngram_range for bigrams
vectorizer = CountVectorizer(ngram_range=(2, 2))

# Fit and Transform the corpus
X = vectorizer.fit_transform(corpus)

# Display the Vocabulary
print("Vocabulary:\n", vectorizer.get_feature_names_out(),"\n")

# Display the BoW Representation
print("BoW Representation:\n", X.toarray())
```

```
Vocabulary:
['ford introduced', 'ford model', 'henry ford', 'introduced the',
'model ford',  'model was', 'the model', 'was revolutionary']

BoW Representation:
 [[1 1 1 1 1 1 1 1]]
```

# Creating n-grams in Python: NLTK

```python
import nltk
from nltk.tokenize import word_tokenize
from nltk.util import ngrams
from collections import Counter
nltk.download('punkt')

# Example text
text = "Henry Ford introduced the Model T. Ford Model T was revolutionary."

# Tokenize the text
tokens = word_tokenize(text)

# Generate bigrams
bigrams = list(ngrams(tokens, 2))

# Create vocabulary (unique bigrams)
vocabulary = list(set(bigrams))

# Create Bag of Words representation for bigrams
bow_vector = [bigrams.count(bigram) for bigram in vocabulary]

# Display results
print("Vocabulary (Bigrams):\n", vocabulary, "\n")
print("BoW Representation (Bigrams):", bow_vector)
```

# Creating n-grams in Python: NLTK (Continued)

```
Vocabulary (Bigrams):
[(’was’, ’revolutionary’), (’T.’, ’Ford’), (’Ford’, ’Model’),
(’revolutionary’, ’.’), (’Ford’, ’introduced’), (’the’, ’Model’),
(’Model’, ’T.’), (’introduced’, ’the’), (’Model’, ’T’),
(’Henry’, ’Ford’), (’T’, ’was’)]

BoW Representation (Bigrams): [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]
```

# Creating n-grams in Python: SpaCy

```python
import spacy
from collections import Counter
from itertools import tee

# Load SpaCy model
nlp = spacy.load("en_core_web_sm")

# Example text
text = "Henry Ford introduced the Model T. Ford Model T was revolutionary."

# Process the text with SpaCy to create a Doc object
doc = nlp(text)

# Tokenize the text and filter out unwanted tokens
tokens = [token.text for token in doc if not token.is_punct and not token.is_space]

# Function to generate bigrams
def generate_bigrams(tokens):
    iterables = tee(tokens, 2)
    for i, iter in enumerate(iterables):
        for _ in range(i):
            next(iter, None)
    return zip(*iterables)

# Generate bigrams
bigrams = list(generate_bigrams(tokens))

# Create vocabulary (unique bigrams)
vocabulary = list(set(bigrams))

# Create Bag of Words (BoW) representation for bigrams
bow_vector = [bigrams.count(bigram) for bigram in vocabulary]
```

# Creating n-grams in Python: SpaCy (Continued)

```
# Display results
print("Vocabulary (Bigrams):\n", vocabulary, "\n")
print("BoW Representation (Bigrams):", bow_vector)
```

```
Vocabulary:
[’T’, ’was’, ’Ford’, ’introduced’, ’Henry’, ’T.’, ’.’, ’the’,
’revolutionary’, ’Model’]

BoW Representation: [1, 1, 2, 1, 1, 1, 1, 1, 1, 2]
```

# Contents

1. Bag of Words (BoW)
   - BoW Representation
   - Hands-On: BoW in Python
     - Sklearn
     - NLTK
     - SpaCy
   - Limitations of BoW
2. Introduction to n-grams
   - n-grams Overview
   - Creating n-grams in Python: Sklearn, NLTK, and SpaCy
   - Limitations of n-grams
3. Convolutional Neural Networks (CNN)
   - Convolution
   - CNN Layers
     - Convolutional Layers
     - MaxPooling Layer
   - Building CNN in Python

# Limitations of n-grams

- Limitations
  - Data sparsity issues as n increases.
  - Limited by local context and unable to capture long-range dependencies.
  - High storage and computational requirements for large n-grams.
- Alternative
  - Use sequences of tokens in combination with more sophisticated models.
    - ★ Example: Recurrent Neural Networks (RNNs) and Transformers.

# Contents

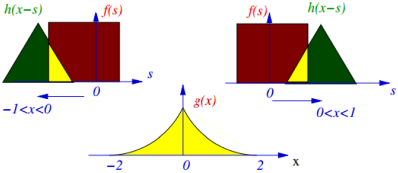
1. Bag of Words (BoW)
   - BoW Representation
   - Hands-On: BoW in Python
     - Sklearn
     - NLTK
     - SpaCy
   - Limitations of BoW
2. Introduction to n-grams
   - n-grams Overview
   - Creating n-grams in Python: Sklearn, NLTK, and SpaCy
   - Limitations of n-grams
3. Convolutional Neural Networks (CNN)
   - Convolution
   - CNN Layers
     - Convolutional Layers
     - MaxPooling Layer
   - Building CNN in Python

# Convolution

Convolution of two functions $f(s)$ and $h(s)$ is defined as follows:

$$(f * h)(x) \doteq \int_{-\infty}^{+\infty} f(s)h(x-s)ds$$

Example:



# Contents

1 Bag of Words (BoW)

- BoW Representation
- Hands-On: BoW in Python
  - Sklearn
  - NLTK
  - SpaCy
- Limitations of BoW

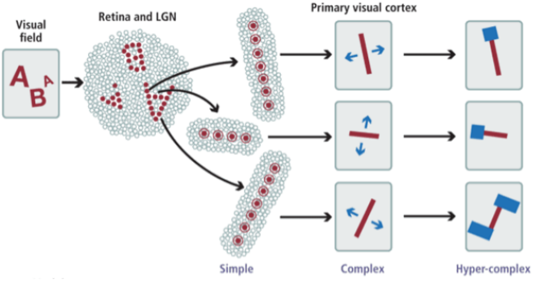
2 Introduction to n-grams

- n-grams Overview
- Creating n-grams in Python: Sklearn, NLTK, and SpaCy
- Limitations of n-grams

3 Convolutional Neural Networks (CNN)

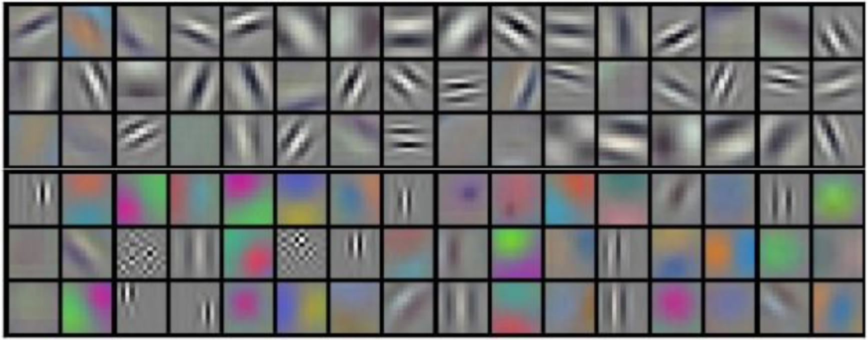
- Convolution
- CNN Layers
  - Convolutional Layers
  - MaxPooling Layer
- Building CNN in Python

# Visual Cortex



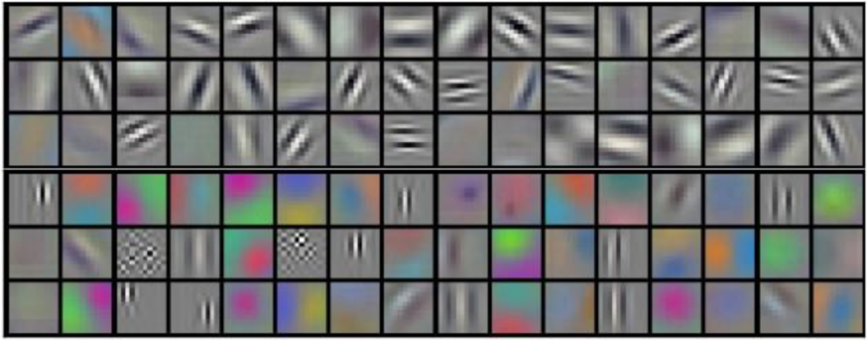
- In the Visual Cortex, simple cells become active when they are subjected to stimuli such as edges.
- Complex cells then combine the information from several simple cells to detect the position and orientation of a structure.
- Hypercomplex cells detect endpoints and crossing lines using this position and orientation information, which is then used in the brain's secondary cortex for information association.

# Convolutional Layers: Filters



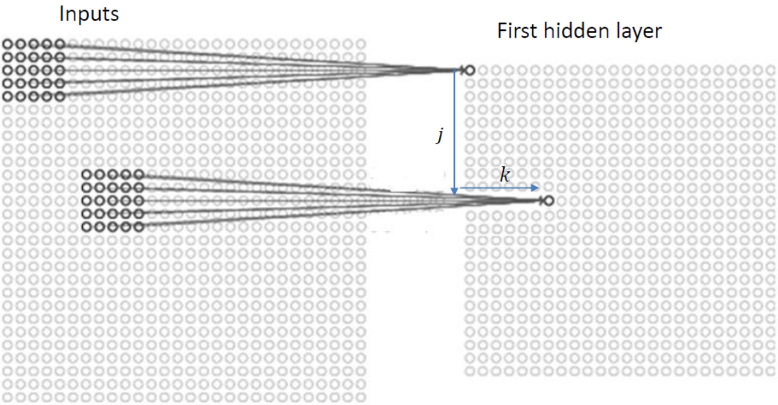
- In early work on CNNs by Krizhevsky et al., they used 96 filters of size $11 \times 11$. Each filter detected certain features such as horizontal edges, vertical edges, slanted edges, etc.
- It eventually became apparent that it is not necessary to preselect the filters. Instead, the neural network training process can determine the shape of the filters.

# Convolutional Layers: Filters

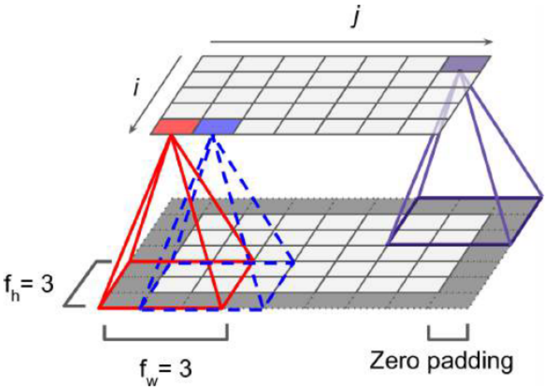


- In early work on CNNs by Krizhevsky et al., they manually designed 96 filters of size $11 \times 11$. Each filter detects certain feature: horizontal edge, vertical edge, slanted edge, etc.
- Eventually, it became apparent that it is not necessary to preselect the filters, but that we could let neural network training process determine the shape of those filters.

# Convolutional Layers: Padding



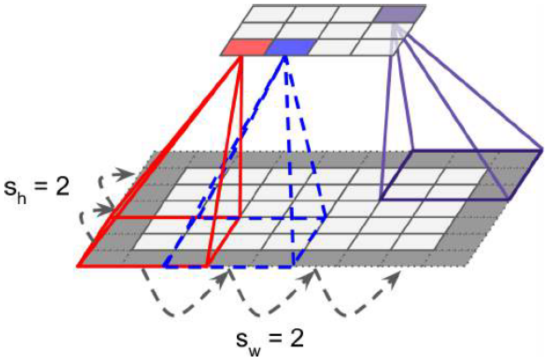
# Convolutional Layers: Padding



Keras: padding can be set to either 'SAME' or 'VALID'

Source: *Hands-On Machine Learning with Scikit-Learn and TensorFlow* by A. Géron

# Convolutional Layers: Strides



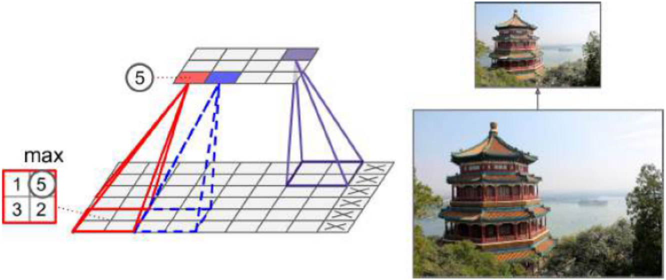
Keras: strides can be set to $(s_w, s_h)$

Source: *Hands-On Machine Learning with Scikit-Learn and TensorFlow* by A. Géron

# MaxPooling Layer

Example:

Max pooling layer with $2 \times 2$ filter, strides$= 2$, and padding=‘VALID’



Source: *Hands-On Machine Learning with Scikit-Learn and TensorFlow* by A. Géron

# Contents

1. Bag of Words (BoW)
   - BoW Representation
   - Hands-On: BoW in Python
     - Sklearn
     - NLTK
     - SpaCy
   - Limitations of BoW
2. Introduction to n-grams
   - n-grams Overview
   - Creating n-grams in Python: Sklearn, NLTK, and SpaCy
   - Limitations of n-grams
3. Convolutional Neural Networks (CNN)
   - Convolution
   - CNN Layers
     - Convolutional Layers
     - MaxPooling Layer
   - Building CNN in Python

# Building CNN in Python

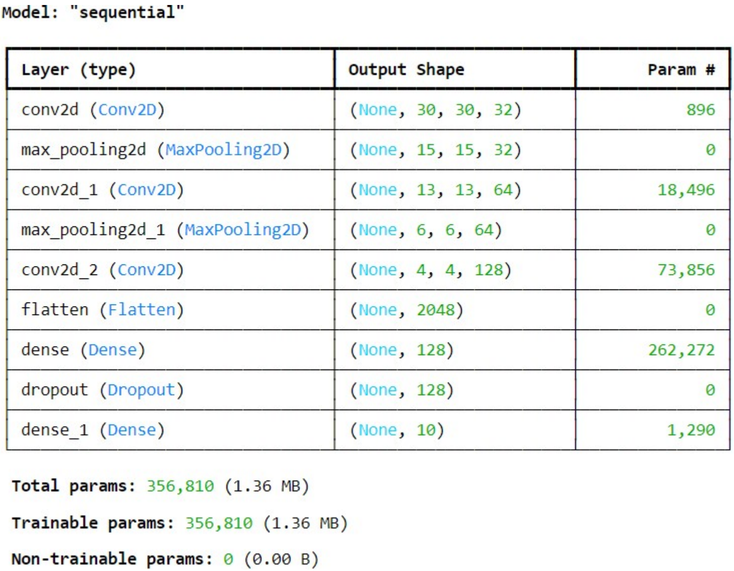
```
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout

# Build the CNN model
model = Sequential([
    Conv2D(32, (3, 3), activation=’relu’, input_shape=(32, 32, 3)),
    MaxPooling2D((2, 2)),
    Conv2D(64, (3, 3), activation=’relu’),
    MaxPooling2D((2, 2)),
    Conv2D(128, (3, 3), activation=’relu’),
    Flatten(),
    Dense(128, activation=’relu’),
    Dropout(0.5),
    Dense(10, activation=’softmax’)
])

# Compile the model
model.compile(optimizer=’adam’, loss=’categorical_crossentropy’, metrics=[’accuracy’])

model.summary()
```

# Building CNN in Python (Continued)



```
Model: "sequential"

┏━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━┓
┃ Layer (type)                    ┃ Output Shape           ┃       Param # ┃
┡━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━┩
│ conv2d (Conv2D)                 │ (None, 30, 30, 32)     │           896 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ max_pooling2d (MaxPooling2D)    │ (None, 15, 15, 32)     │             0 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ conv2d_1 (Conv2D)               │ (None, 13, 13, 64)     │        18,496 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ max_pooling2d_1 (MaxPooling2D)  │ (None, 6, 6, 64)       │             0 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ conv2d_2 (Conv2D)               │ (None, 4, 4, 128)      │        73,856 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ flatten (Flatten)               │ (None, 2048)           │             0 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ dense (Dense)                   │ (None, 128)            │       262,272 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ dropout (Dropout)               │ (None, 128)            │             0 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ dense_1 (Dense)                 │ (None, 10)             │         1,290 │
└─────────────────────────────────┴────────────────────────┴───────────────┘

 Total params: 356,810 (1.36 MB)

 Trainable params: 356,810 (1.36 MB)

 Non-trainable params: 0 (0.00 B)
```