# Q.1: Describe the following in detail: CNN.

→ A Convolutional Neural Network (CNN) is a specialised type of artificial neural network.

### Key Concepts of CNNs

1. **Architecture Overview:**
   → Convolution layer: extract features using convolution operations.
   → Pooling layer: Reduce dimensionality while retaining critical information.
   → Full connected layer: Map extracted features to the output.
2. **Key Operations in CNNs**
   *Convolution:*

   - This operation applies filters (kernels) to input data to extract specific features like edges, textures, or patterns.
   - Filters slide across the input (a process called “stride”), creating feature maps.
   - Each filter learns to detect a specific feature during training.

   *Pooling:*

   - Pooling reduces the spatial dimensions of the features maps, improving computational efficiency and reducing overfitting.
     → **Max polling:** Selects the maximum value in a region.
     → **Average pooling:** Computes the average value in a region.

   *Full connected layers:*
   - These layers flatten the feature maps into a 1D vector and connect them to output neurons.
   - Used for final tasks like classification or regression.

### Key Examples of CNN Models:

1. **LeNet (1998)**: Among the first CNNs, designed for digit recognition.
2. **AlexNet (2012)**: Popularized deep CNNs by winning the ImageNet challenge.
3. **VGGNet (2014)**: Demonstrated the power of deeper architectures.
4. **ResNet (2015)**: Introduced skip connections to address vanishing gradients in deep networks.

# Q.2: Choose the correct answer:

**1. In a CNN, what does the term "stride" refer to?**
a) The size of the convolutional kernel.

b) The number of convolutional layers.
c) The number of training epochs.
d) The step size for moving the kernel across input data.

**2. Which type of pooling operation selects the maximum value from each region?**
a) Average Pooling
b) Global Max Pooling
c) Max Pooling
d) Min Pooling

**3. What is the purpose of the padding technique in CNNs?**
a) Increase model complexity.
b) Improve computational efficiency.
c) Preserve spatial dimensions.
d) Enhance feature extraction.

**4. What role do convolutional layers play in CNNs?**
a) Introduce non-linearity.
b) Extract Feature.
c) Aggregate predictions
d) Reduce overfitting.

**5. What is the primary purpose of pooling layers in CNNs?**
a) Increase model complexity
b) Reduce spatial dimensions of feature maps
c) Apply activation functions
d) Perform convolution operations

**6- Which layer in a CNN is responsible for introducing non-linearity into the model?**
a) Convolution Layer.
b) Pooling Layer.
c) FC Layer.
d) Activation Function Layer.

**7. What is What role do fully connected layers play in a CNN?**
a) They extract features from the input data
b) They reduce spatial dimensions of feature maps.
c) They aggregate learned features and make predictions.
d) They apply non-linear transformations to feature maps.

**8- What is the primary advantage of using Convolutional Neural Networks (CNNs) over Artificial Neural Networks (ANNs) for computer vision tasks?**
a) CNNs require less training data.
b) CNNs can automatically detect important features from data.

c) CNNs are simpler to implement.
d) CNNs are faster to train

## Q3. Write and discuss the python program to extract six types of feature engineering for any long text then enter these features to DL.

1. **Text Preprocessing**: Tokenization, removal of special characters, and optional stemming/lemmatization.
2. **Feature Extraction:** Week-end extract the following features;

- Word Count
- Unique Word Count
- Average Word Length
- TF-IDF (Term Frequency-Inverse Document Frequency)
- Sentiment Score
- Topic Modeling (LDA - Latent Dirichlet Allocation)

3. **Deep Learning Integration**: Combine these features into a single input vector for use in a DL model.

```
import re
import numpy as np
import pandas as pd
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.decomposition import LatentDirichletAllocation
from textblob import TextBlob
from keras.models import Sequential
from keras.layers import Dense

# Function to preprocess text
def preprocess_text(text):
    # Convert to lowercase
    text = text.lower()
    # Remove special characters
    text = re.sub(r'[^a-zA-Z\s]', '', text)
    # Tokenize text
    tokens = text.split()
    return tokens

# Function to extract features
def extract_features(text):
    tokens = preprocess_text(text)

    # 1. Word Count
```

```
    word_count = len(tokens)

    # 2. Unique Word Count
    unique_word_count = len(set(tokens))

    # 3. Average Word Length
    avg_word_length = np.mean([len(word) for word in tokens]) if tokens
else 0

    # 4. TF-IDF Features
    tfidf_vectorizer = TfidfVectorizer(max_features=10)
    tfidf_matrix = tfidf_vectorizer.fit_transform([text])
    tfidf_features = tfidf_matrix.toarray().flatten()

    # 5. Sentiment Score
    sentiment = TextBlob(text).sentiment
    sentiment_score = sentiment.polarity  # Polarity ranges from -1 to 1

    # 6. Topic Modeling Features (using LDA)
    lda = LatentDirichletAllocation(n_components=2, random_state=42)
    tfidf_for_lda = tfidf_vectorizer.fit_transform([text])
    lda_features = lda.fit_transform(tfidf_for_lda).flatten()

    # Combine features into a single vector
    combined_features = [
        word_count,
        unique_word_count,
        avg_word_length,
        sentiment_score,
        *tfidf_features,
        *lda_features
    ]
    return np.array(combined_features)

# Sample long text
sample_text = """
Deep learning has transformed the field of artificial intelligence. Its
applications include natural language processing,
image recognition, and game playing. CNNs and RNNs are widely used for
tasks requiring spatial and sequential data processing.
"""

# Extract features from the text
features = extract_features(sample_text)

# Normalize features for deep learning
features = np.array(features).reshape(1, -1)
```

```python
# Deep Learning Model (Simple Example)
model = Sequential([
    Dense(16, input_dim=features.shape[1], activation='relu'),
    Dense(8, activation='relu'),
    Dense(1, activation='sigmoid')  # Binary classification output
])

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

# Print feature vector and model summary
print("Extracted Features:", features)
model.summary()
```

```
>>>Extracted Features: [[34.         31.          6.17647059 -0.2
0.65465367  0.43643578
   0.21821789  0.21821789  0.21821789  0.21821789  0.21821789  0.21821789
   0.21821789  0.21821789  0.82499577  0.17500423]]
```

**Model: "sequential"**

| Layer (type) | Output Shape | Param # |
|---|---|---|
| dense (Dense) | (None, 16) | 272 |
| dense_1 (Dense) | (None, 8) | 136 |
| dense_2 (Dense) | (None, 1) | 9 |

**Total params:** 417 (1.63 KB)

**Trainable params:** 417 (1.63 KB)

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