School Connect Online

SCO INTERNATIONAL OLYMPIAD

CLASS 10 AI SAMPLE QUESTION PAPER

SCO International AI Olympiad

A carefully formatted question paper for schools, teachers, parents, and students, with answer keys, explanations, and visual learning aids for AI, machine learning, NLP, computer vision, and responsible AI.

Designed from Class 10 AI pathways and aligned with SCO's platform flow for guided preparation, practice, reporting, and future-ready academic growth.

- exam-ready AI concepts covering machine learning, NLP, computer vision, ethics, and Python AI tools
- case-study based reasoning for healthcare, autonomous systems, recruitment, language translation, and surveillance
- visual prompts placed inside question blocks to support understanding and revision before PDF publication

ML Metrics	Clustering	NLP	Transformers	Fairness
Python AI	OpenCV	CNN	Case Study	Ethics

Name:.....

Registration ID:..... Contact No.:.....

SCO INTERNATIONAL AI OLYMPIAD

Class 10 | Question Paper Set S | 2025-26

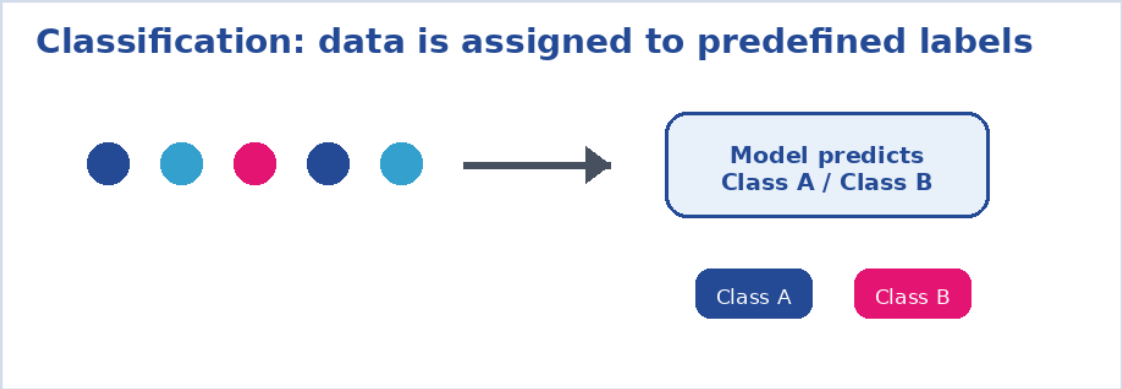
Paper Structure: 50 multiple-choice questions with one correct answer each. The paper includes AI concepts, NLP, ethics/fairness, Python code reasoning, OpenCV, CNNs, TensorFlow/Keras, and case-study based reasoning.

Section A: Core AI, Machine Learning and NLP Foundations

1.

General Concept

Which of the following best describes classification in machine learning?



A) A process that predicts a continuous value.

B) A process that assigns predefined labels to data.

C) A technique used solely for data clustering.

D) A method to compress data into smaller representations.

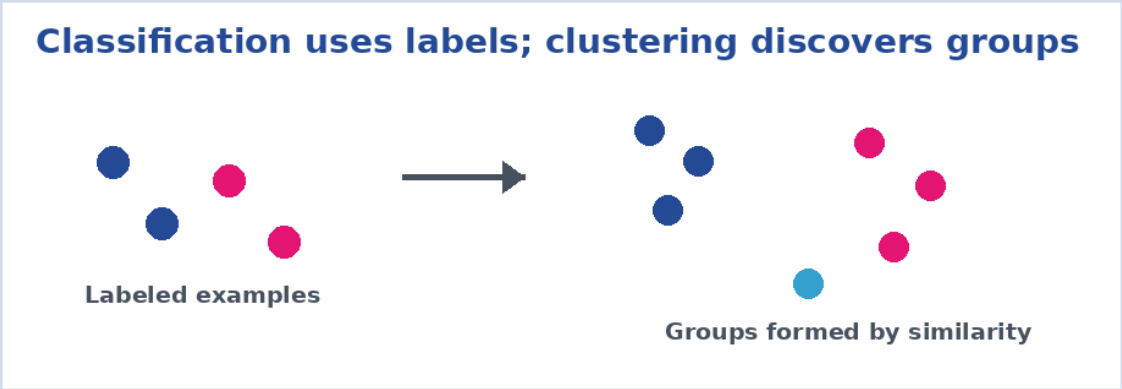
Answer: **B**

Explanation: Classification is a supervised learning technique where data is assigned to predefined categories (labels), as opposed to predicting continuous values (which is regression).

2.

General Concept

What is the main difference between classification and clustering?



A) Classification is unsupervised, while clustering is supervised.

B) Classification uses labeled data; clustering groups data without prior labels.

C) Classification groups data based on similarity; clustering assigns fixed labels.

D) Both use the same algorithms, but clustering is faster.

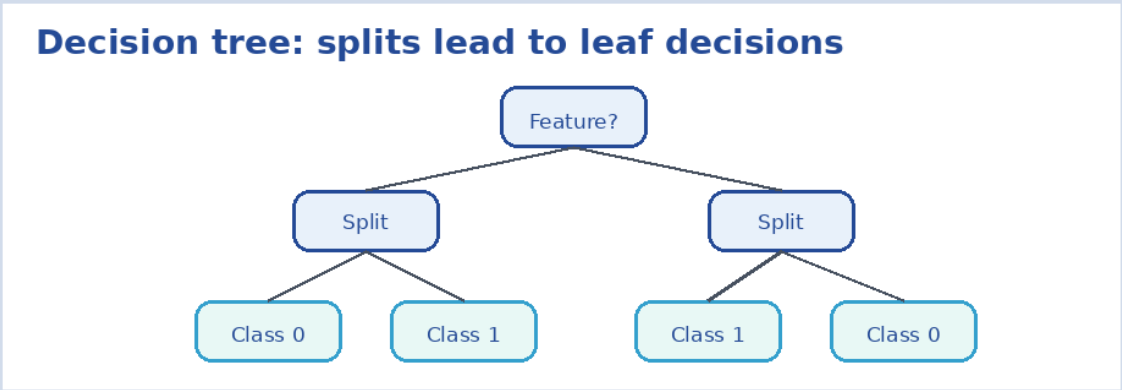
Answer: B

Explanation: Classification uses labeled training data to predict categories for new data, while clustering is an unsupervised technique that groups similar data points without pre-assigned labels.

3.

General Concept

In a decision tree, what is the role of a leaf node?



A) It is where the data is split into smaller subsets.

B) It represents the final decision or classification outcome.

C) It stores the entire dataset for future processing.

D) It acts as an intermediate step in calculating the average.

Answer: B

Explanation: Leaf nodes are the terminal nodes of a decision tree that contain the final output (classification or decision)

after all splits have been made.

4.

General Concept

Which of the following measures is commonly used to determine the best split at each node in a decision tree?

A) Euclidean distance

B) Information Gain

C) Mean Squared Error

D) Clustering Coefficient

Answer: B

Explanation: Information Gain (or its alternatives like Gini impurity) is used to measure how well a feature splits the data into classes, with higher values indicating better splits.

5.

General Concept

What does it mean when a classification model is overfitting?

A) The model performs poorly on both training and new data.

B) The model is too simple and cannot capture data patterns.

C) The model performs well on training data but poorly on new data.

D) The model ignores the input features completely.

Answer: C

Explanation: Overfitting occurs when a model memorizes the training data, resulting in high accuracy on the training set but poor generalization to unseen data.

6.

General Concept

Which real-world scenario best illustrates the use of clustering?

A) Sorting emails into “spam” and “not spam” folders.

B) Grouping customers based on their purchasing habits without predefined labels.

C) Predicting the price of a house based on its features.

D) Automatically translating text from one language to another.

Answer: B

Explanation: Clustering groups similar items together without predefined labels. Grouping customers by purchasing habits is a common unsupervised clustering application.

7.

General Concept

Why is human language considered more challenging for computers to understand compared to computer language?

A) Human language is entirely unstructured with no rules.

B) Human language is highly ambiguous and context-dependent.

C) Computer language is designed for non-logical reasoning.

D) Computers have limited memory to store human language.

Answer: B

Explanation: Human language is ambiguous, contains idioms, slang, and context that can change meaning, making it much more challenging for computers to process compared to the highly structured nature of computer languages.

8.

General Concept

Which machine learning technique is most commonly used to process sequential data like human language?

A) Convolutional Neural Networks (CNNs)

B) Decision Trees

C) Recurrent Neural Networks (RNNs)

D) K-Means Clustering

Answer: C

Explanation: Recurrent Neural Networks (RNNs) are designed to handle sequential data, such as text or speech, making them suitable for NLP tasks where context and order are important.

9.

General Concept

What is one main advantage of using a Random Forest over a single decision tree?

A) Random Forests are always faster to train.

B) Random Forests reduce overfitting by averaging multiple trees.

C) Random Forests do not require any data preprocessing.

D) Random Forests use unsupervised learning to classify data.

Answer: B

Explanation: A Random Forest builds multiple decision trees and averages their results, which reduces the likelihood of overfitting compared to a single decision tree.

10.

General Concept

Which of the following is a major challenge when building AI systems that process human language?

A) Lack of data available for training

B) Overabundance of strictly structured language rules

C) Variability in dialects, accents, and informal expressions

D) Excessive computational power making systems too fast

Answer: C

Explanation: Human language varies greatly across dialects, accents, and informal expressions, which makes it challenging for AI systems to accurately understand and interpret the intended meaning.

Section B: NLP, Data Processing and Model Evaluation

11.

NLP/Data Processing

In an NLP project, you are processing raw text data from news articles. One of the first steps is tokenization, where the text is split into individual tokens (words, punctuation, etc.). Consider the following pseudo-code for tokenization:

```
import nltk
```

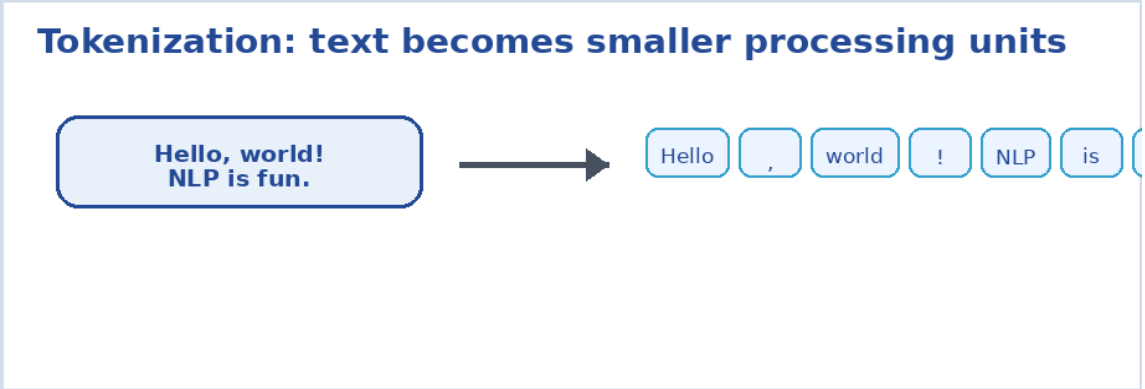
nltk.download('punkt')

```
from nltk.tokenize import word_tokenize
```

```
text = "Dr. Smith's lecture on NLP, held at 10:00 a.m., was very enlightening!"
tokens = word_tokenize(text)
print(tokens)
```

Question:

Which of the following best describes the challenges faced during tokenization in this scenario?



A) Handling abbreviations, punctuation, and numerical expressions correctly

B) Converting all tokens into their base forms (stemming)

C) Removing stop words from the text

D) Translating the text from one language to another

Answer: A

Explanation: Tokenization must correctly handle abbreviations (e.g., "Dr."), punctuation (commas, apostrophes), and numerical expressions (e.g., "10:00 a.m."). Option B is related to stemming, C to stop word removal, and D to translation—each a separate processing step.

12.

NLP/Data Processing

A research team is building a system to extract key information from legal documents. After tokenization, the system applies part-of-speech (POS) tagging to assign grammatical categories (nouns, verbs, adjectives) to each token.

Question:

Why is accurate POS tagging critical in syntactic analysis for legal document processing?

A) It helps convert legal jargon into simpler language.

B) It enables the extraction of meaningful relationships and entities by understanding sentence structure.

C) It automatically summarizes the entire document.

D) It encrypts the document for secure processing.

Answer: **B**

Explanation: POS tagging provides grammatical context that is essential for parsing sentence structure and identifying relationships among entities (e.g., subjects, objects). This is crucial in legal documents where precision is paramount. Options A, C, and D are not direct benefits of POS tagging.

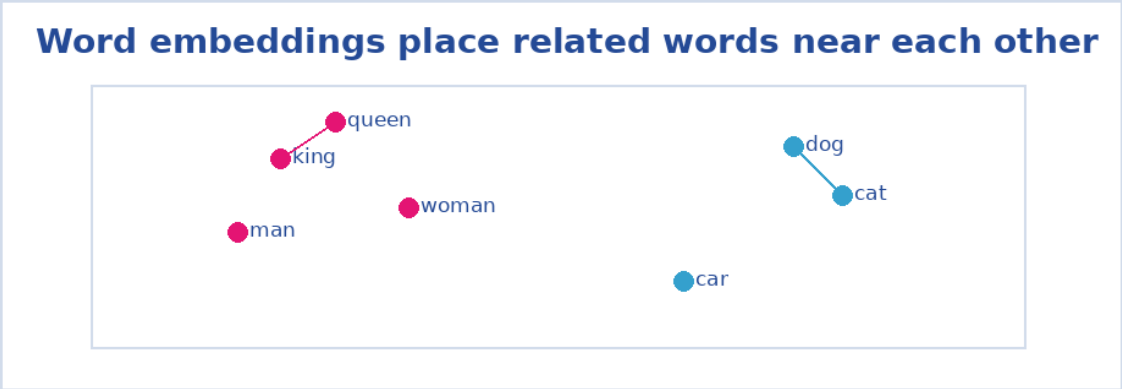
13.

NLP/Data Processing

To improve its chatbot's understanding of user queries, a company integrates word embeddings using the Word2Vec algorithm. This maps words into a continuous vector space where semantically similar words are closer together.

Question:

What is the primary advantage of using word embeddings in semantic analysis?



A) They provide a one-hot encoded representation of words.

B) They reduce the dimensionality of text data to a single scalar value.

C) They capture contextual and semantic relationships between words, enabling similarity comparisons.

D) They enforce strict rules for grammar correction.

Answer: **C**

Explanation: Word embeddings capture semantic relationships and contextual similarities by representing words as dense vectors in a continuous space. This allows the model to understand similarities between words like "king" and "queen." Option A describes one-hot encoding, which is sparse and less effective at capturing semantics. Option B misrepresents dimensionality, and D is not a function of embeddings.

14.

NLP/Data Processing

An AI system is designed to interpret customer support emails. However, the system struggles with ambiguous words (e.g., "bank" can mean a financial institution or the side of a river). The developers

decide to incorporate word-sense disambiguation techniques.

Question:

Which approach is most effective for resolving ambiguity in natural language processing?

A) Relying solely on keyword matching

B) Incorporating context through surrounding words using algorithms like Lesk or neural models

C) Ignoring ambiguous words during processing

D) Converting all words to their plural forms

Answer: B

Explanation: Effective word-sense disambiguation requires context. Algorithms like the Lesk algorithm or modern neural methods (e.g., BERT) use surrounding words to determine the correct sense of ambiguous words. Option A is too simplistic, C discards important data, and D is unrelated.

15.

NLP/Data Processing

A data scientist is building an AI system to analyze social media posts. The preprocessing pipeline includes lowercasing, tokenization, stop-word removal, and stemming. Consider the following pseudo-code:

```
import nltk
from nltk.tokenize import word_tokenize
from nltk.corpus import stopwords
from nltk.stem import PorterStemmer
```

nltk.download('punkt')

nltk.download('stopwords')

```
ps = PorterStemmer()
```

stop_words = set(stopwords.words('english'))

```
def preprocess(text):
    tokens = word_tokenize(text.lower())
    filtered = [word for word in tokens if word not in stop_words]
    stemmed = [ps.stem(word) for word in filtered]
    return stemmed
```

```
print(preprocess("Running runners ran rapidly towards the finish line."))
```

Question:

What is the primary goal of this text preprocessing pipeline?

A) To remove noise and standardize the text for improved model performance

B) To convert all text into a numeric format for deep learning

C) To translate the text into another language

D) To detect the sentiment of the text directly

Answer: **A**

Explanation: The preprocessing pipeline cleans and standardizes text data by converting it to lowercase, removing stop words, and applying stemming. This reduces noise and variability, which enhances model performance in downstream tasks. Option B is part of vectorization, C is translation, and D is sentiment analysis.

16.

NLP/Data Processing

A company uses sentiment analysis to gauge customer opinions from product reviews. The system uses a combination of tokenization, POS tagging, and a pre-trained sentiment classifier (e.g., using LSTM).

Question:

Which of the following is a significant challenge in building an effective sentiment analysis model?

A) Distinguishing between positive and negative sentiments when subtle context or sarcasm is present

B) Converting numerical values into words

C) Extracting stop words from the dataset

D) Automatically generating product images from reviews

Answer: **A**

Explanation: Subtle contexts and sarcasm are common challenges in sentiment analysis. Models must understand nuanced language to accurately classify sentiment. Options B, C, and D do not address the core difficulty in sentiment interpretation.

17.

NLP/Data Processing

A project requires the extraction of named entities (people, organizations, locations) from a collection of news articles. Using NLTK, a developer writes:

```
import nltk
```

nltk.download('maxent_ne_chunker')

nltk.download('words')

```
from nltk import pos_tag, ne_chunk, word_tokenize
```

```
text = "Barack Obama was born in Hawaii and served as the president of the United States."
tokens = word_tokenize(text)
```

pos_tags = pos_tag(tokens)

tree = ne_chunk(pos_tags)

```
print(tree)
```

Question:

What is the purpose of the ne_chunk() function in this code?

A) To tokenize the text into sentences

B) To assign part-of-speech tags to tokens

C) To recognize and classify named entities in the text

D) To remove punctuation from the text

Answer: C

Explanation: The ne_chunk() function in NLTK takes the POS-tagged tokens and identifies named entities, classifying them into categories such as PERSON, ORGANIZATION, or LOCATION. Options A, B, and D describe earlier steps in the pipeline.

18.

NLP/Data Processing

A developer wants to build a simple Recurrent Neural Network (RNN) for next-word prediction using TensorFlow. Consider the following simplified code snippet:

```
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import SimpleRNN, Dense, Embedding
```

```
model = Sequential([
    Embedding(input_dim=5000, output_dim=64, input_length=20),
    SimpleRNN(128),
    Dense(5000, activation='softmax')
```

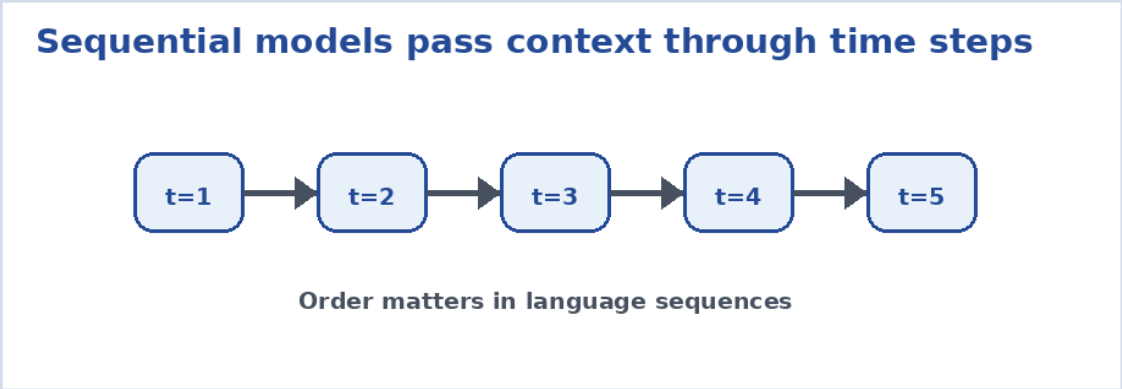
])

model.compile(optimizer='adam', loss='sparse_categorical_crossentropy')

model.summary()

Question:

What is the primary function of the Embedding layer in this RNN model?



A) To convert words into a one-hot encoded vector

B) To reduce the sequence length of the input

C) To map words into continuous, dense vector representations that capture semantic meaning

D) To output a sequence of predictions directly

Answer: **C**

Explanation: The Embedding layer transforms integer-encoded words into dense vector representations, which capture semantic relationships between words. This is essential for enabling the RNN to learn language patterns. Option A describes one-hot encoding (less efficient), B is unrelated, and D is the role of the Dense layer.

19.

NLP/Data Processing

After training a text classification model, a data scientist needs to evaluate its performance. They use accuracy, precision, recall, and F1 score to measure the model's effectiveness on a test set.

Question:

Why is it important to consider both precision and recall, rather than only accuracy, in evaluating NLP models?

A) Because accuracy only measures the speed of the model

B) Because precision and recall provide a more detailed view of model performance, particularly when classes are imbalanced

C) Because precision and recall are easier to compute

D) Because accuracy is not relevant for text data

Answer: **B**

Explanation: Accuracy may be misleading, especially in cases of class imbalance. Precision measures the correctness of positive predictions, while recall measures how many actual positives were captured. Together, they provide a comprehensive picture of model performance, especially when the cost of false positives and false negatives differs.

20.

NLP/Data Processing

In preparing text data for an AI data processing program, a team applies techniques such as TF-IDF (Term Frequency-Inverse Document Frequency) to convert text into a numerical representation. This transformation helps capture the importance of words within documents.

Question:

What is the key advantage of using TF-IDF over simple word counts in NLP feature engineering?

A) TF-IDF automatically translates text to multiple languages

B) TF-IDF reduces the size of the dataset by removing stop words

C) TF-IDF weights words by their importance, reducing the influence of common but less informative words

D) TF-IDF converts text into a binary format for neural networks

Answer: C

Explanation: TF-IDF adjusts the raw frequency of words by how often they appear across all documents. This helps emphasize unique words that are more informative for the task, rather than common words that occur in nearly every document. Options A, B, and D do not capture the primary purpose of TF-IDF.

Section C: Responsible AI, Fairness and Ethics

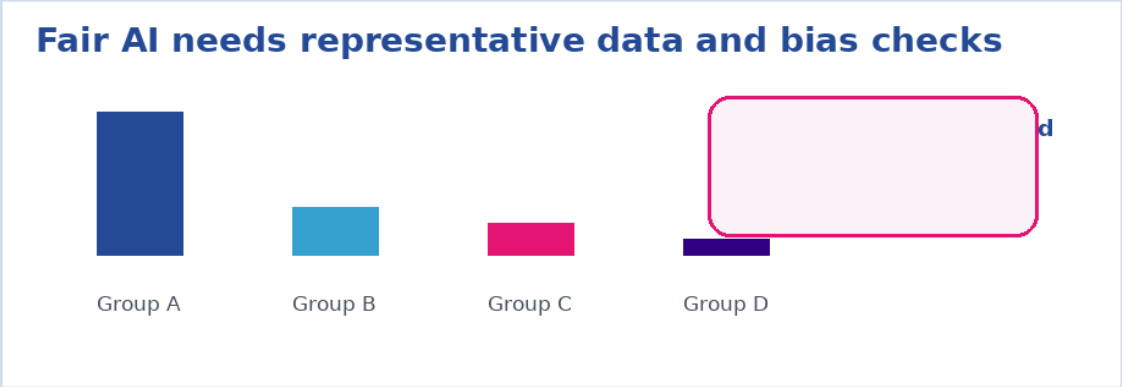
21.

Ethics & Fairness

A metropolitan police department deployed a facial recognition system to assist in identifying suspects. Shortly after deployment, community feedback revealed that the system had a higher error rate when identifying individuals with darker skin tones. Investigations revealed that the training data lacked sufficient diversity.

Question:

What is the most effective initial step to mitigate this bias?



A) Switch to a different algorithm that inherently does not show bias.

B) Increase the diversity of the training dataset by including more images of underrepresented groups.

C) Lower the confidence threshold for identification across all groups.

D) Implement manual verification for all identifications.

Answer: B

Explanation: The root cause of bias here is an imbalanced training dataset. By increasing diversity, the model can learn to recognize faces more accurately across different skin tones. Changing the algorithm or thresholds doesn't address the underlying data imbalance.

22.

Ethics & Fairness

A large company implemented an AI system to screen resumes. It was later discovered that the system systematically downgraded applications from candidates belonging to certain minority groups because the historical hiring data used for training reflected past human biases.

Question:

Which intervention would best address the ethical issue in this hiring AI system?

A) Removing demographic features from the dataset completely.

B) Retraining the model on a dataset that has been carefully balanced and debiased to reflect fair candidate profiles.

C) Reducing the number of resumes processed by the algorithm.

D) Increasing the weight of technical skills in the model's decision-making process.

Answer: B

Explanation: Retraining on a balanced dataset that has been audited and adjusted for bias is key to mitigating inherited biases from historical data. Removing demographic features might help, but it does not guarantee fairness if other proxies for demographics remain.

23.

Ethics & Fairness

A fintech company uses an AI system to automate loan approvals. Analysis reveals that applicants from lower-income neighborhoods are being denied loans at a significantly higher rate. Investigation shows that historical lending data, which reflected socioeconomic disparities, was used to train the model.

Question:

What measure is most likely to reduce bias in the loan approval system?

A) Incorporate additional socioeconomic factors to refine risk assessments while ensuring ethical review of feature selection.

B) Exclude geographic data from the model entirely.

C) Increase the loan approval rate uniformly across all areas.

D) Switch to a rule-based system instead of AI.

Answer: A

Explanation: Adding relevant socioeconomic factors and rebalancing the training data can help the model learn a more equitable representation of risk. Excluding geographic data might remove some bias but could also discard useful information, while a uniform approval rate ignores individual risk.

24.

Ethics & Fairness

A global company deploys a sentiment analysis tool to monitor social media sentiment. However, the tool consistently misinterprets posts from certain cultural regions due to nuances in language and idiomatic expressions that were underrepresented in the training data.

Question:

Which strategy would best improve the fairness of the sentiment analysis model?

A) Incorporate additional training data from underrepresented cultural groups and fine-tune the model accordingly.

B) Use a simpler sentiment analysis model to avoid overfitting.

C) Remove slang and idiomatic expressions from the input data.

D) Apply uniform threshold adjustments for all sentiment classifications.

Answer: A

Explanation: Expanding the training data to include more culturally diverse expressions will enable the model to better understand and interpret sentiment across different regions. Simplifying the model or applying thresholds does not address the root cause of cultural bias.

25.

Ethics & Fairness

An AI diagnostic tool is used to analyze medical images for disease detection. It was observed that the tool performs less accurately for images from certain demographic groups. Further analysis indicated that these groups were underrepresented in the training dataset.

Question:

What is the most ethically responsible action to improve diagnostic fairness?

A) Retrain the model using a more diverse set of medical images that adequately represent all demographics.

B) Increase the number of layers in the neural network.

C) Standardize all images to the same resolution.

D) Apply a post-processing correction factor for underrepresented groups.

Answer: A

Explanation: A diverse training dataset is essential to ensure that the AI diagnostic tool performs well across all demographic groups. Increasing network complexity or standardizing resolution won't resolve the data representation issue.

26.

Ethics & Fairness

A manufacturer of autonomous vehicles is developing algorithms for emergency decision-making (e.g., in unavoidable accident scenarios). Ethical dilemmas arise when the algorithm must choose between actions that might prioritize passenger safety over pedestrian safety.

Question:

Which ethical framework is most suitable to guide the decision-making process in this scenario?

A) Random decision-making to avoid systematic bias.

B) Utilitarian ethics, which seeks to minimize overall harm.

C) Strict adherence to a pre-programmed rule regardless of context.

D) Maximizing passenger safety at all costs.

Answer: B

Explanation: Utilitarian ethics, which focus on minimizing overall harm, is commonly proposed for complex decision-making in autonomous vehicles. It aims to balance conflicting ethical outcomes, unlike fixed rules or random decisions that might not account for overall impact.

27.

Ethics & Fairness

A multinational e-commerce platform uses an AI chatbot to assist customers in multiple languages. Customers from non-English-speaking regions report that the chatbot often misunderstands their queries, leading to poor service experiences. Analysis shows that the chatbot's training data was predominantly in English.

Question:

What is the most effective solution to mitigate language bias in this chatbot system?

A) Limit the chatbot's functionality to English only.

B) Expand the training dataset to include a balanced representation of multiple languages and dialects.

C) Use translation software to convert all inputs to English.

D) Implement a simple rule-based system for non-English queries.

Answer: B

Explanation: Expanding the training dataset to include multiple languages ensures that the chatbot can handle diverse linguistic inputs more accurately. Relying solely on translation or limiting functionality does not address the underlying issue of representation in the training data.

28.

Ethics & Fairness

A city adopts a predictive policing system that uses historical crime data to allocate police resources. Critics argue that the system disproportionately targets minority neighborhoods due to biased historical data. The system's predictions reinforce existing disparities in law enforcement.

Question:

Which of the following actions would most effectively address bias in predictive policing?

A) Increase the number of police officers in all neighborhoods.

B) Audit and adjust the historical data to reduce bias before retraining the model.

C) Remove all historical data and start the model from scratch.

D) Implement a random allocation of police resources regardless of data.

Answer: B

Explanation: Auditing and adjusting historical data to correct biases can help ensure that the predictive policing system does not reinforce existing disparities. Removing data entirely or random allocation ignores valuable information and context.

29.

Ethics & Fairness

A recruitment platform uses an AI-powered chatbot to screen candidates and schedule interviews. Over time, the platform observes that certain demographic groups are less likely to be scheduled for interviews, indicating potential bias in the screening process, possibly due to language or phrasing differences in responses.

Question:

What approach should be taken to evaluate and mitigate bias in this chatbot system?

A) Increase the number of scheduled interviews for all candidates.

B) Conduct a fairness audit of the chatbot's decision-making process and retrain it on a more balanced dataset that includes diverse linguistic expressions.

C) Remove all natural language inputs from the process.

D) Rely solely on manual review after the chatbot interaction.

Answer: B

Explanation: A fairness audit can identify specific biases in the chatbot's responses. Retraining on a balanced dataset

that represents linguistic diversity will help mitigate these biases, rather than bypassing or manually reviewing every case, which is less scalable.

30.

Ethics & Fairness

A financial institution uses an AI system to calculate credit scores. It is discovered that the system disproportionately assigns lower scores to individuals from certain zip codes, which correlates with socioeconomic factors. This raises concerns about fairness and discrimination.

Question:

Which of the following is a recommended strategy to reduce algorithmic bias in credit scoring?

A) Exclude zip code data from the model completely.

B) Reevaluate and adjust the feature weights, ensuring that sensitive attributes are either removed or carefully balanced by incorporating fairness constraints.

C) Increase the overall credit score for all individuals from affected areas.

D) Replace the AI system with a manual credit scoring process.

Answer: **B**

Explanation: By reevaluating and adjusting feature weights and incorporating fairness constraints, the model can be made less sensitive to proxies for discrimination such as zip codes. Simply excluding data may lose valuable information, while blanket adjustments or reverting to manual processes are not scalable or systematic solutions.

Section D: Python, NLP Code and Data Handling

31.

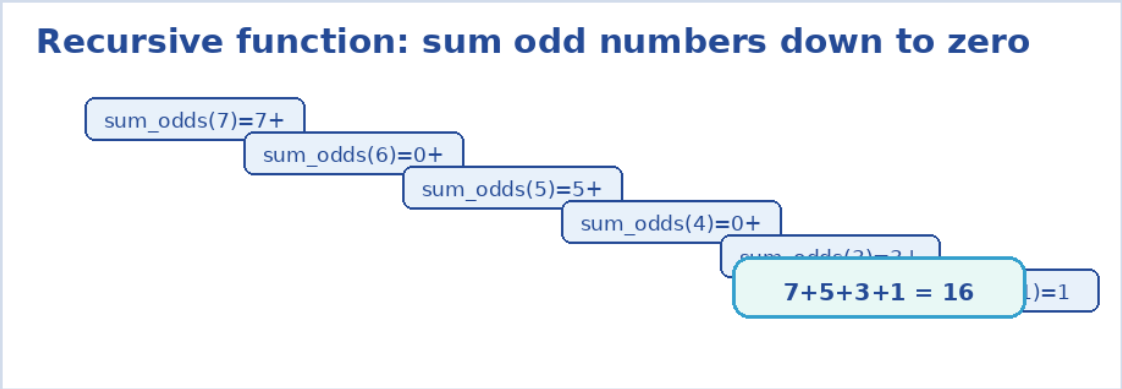
Python/Code Output

```
def sum_odds(n):
    if n <= 0:
        return 0
    else:
        return (n if n % 2 == 1 else 0) + sum_odds(n-1)
```

```
print(sum_odds(7))
```

Question:

What is the output of this code?



A) 16

B) 15

C) 12

D) 10

Answer: A

Explanation: If n = 7: 7 (odd) + sum_odds(6) n = 6: 6 is even → 0 + sum_odds(5) n = 5: 5 (odd) + sum_odds(4) n = 4: even → sum_odds(3) n = 3: 3 (odd) + sum_odds(2) n = 2: even → sum_odds(1) n = 1: 1 (odd) + sum_odds(0) n = 0: returns 0 Thus: 7 + 5 + 3 + 1 = 16. So the correct answer is A) 16.

32.

Python/Code Output

```
lst = [i for i in range(1, 11) if i % 3 == 0 or i % 4 == 0]
print(sum(lst))
```

Question:

What is the output printed by this code?

A) 30

B) 31

C) 32

D) 33

Answer: A

Explanation: The list includes numbers from 1 to 10 divisible by 3 or 4: 3, 4, 6, 8, 9. Their sum is 3 + 4 + 6 + 8 + 9 = 30.

33.

Python/Code Output

```
words = ["apple", "banana", "cherry"]
result = {word: len(word) for word in words if word[0] == 'b'}
print(result)
```

Question:

What is printed by this code?

A) {'apple': 5, 'banana': 6, 'cherry': 6}

B) {'banana': 6}

C) {'banana': '6'}

D) {}

Answer: **B**

Explanation: Only the word "banana" starts with 'b'. Its length is 6, so the dictionary is {'banana': 6}.

34.

Python/Code Output

```
import nltk
```

nltk.download('punkt')

```
from nltk.tokenize import word_tokenize
```

```
text = "Hello, world! NLP is fun."
tokens = word_tokenize(text)
print(tokens)
```

Question:

What is the expected output?

A) ['Hello world NLP is fun']

B) ['Hello', 'world', 'NLP', 'is', 'fun']

C) ['Hello', ',', 'world', '!', 'NLP', 'is', 'fun', '.']

D) ['hello', 'world', 'nlp', 'is', 'fun']

Answer: **C**

Explanation: The NLTK tokenizer splits punctuation as separate tokens. Therefore, punctuation marks like commas, exclamation points, and periods become individual tokens.

35.

Python/Code Output

```
from nltk import FreqDist
tokens = ['data', 'science', 'data', 'ai', 'machine', 'data']
fdist = FreqDist(tokens)
print(fdist.most_common(2))
```

Question:

What does this code print?

A) [('data', 3), ('science', 1)]

B) [('data', 3), ('ai', 1)]

C) [('science', 1), ('machine', 1)]

D) [('data', 3), ('machine', 1)]

Answer: **A**

Explanation: "data" appears 3 times, while "science", "ai", and "machine" each appear once. The most_common(2) method returns the two highest frequency items. Although there is a tie among the words with frequency 1, "science" appears first in the token list.

36.

Python/Code Output

```
import re
text = "Contact: john.doe@example.com or jane_doe123@company.co.uk"
emails = re.findall(r'\S+@\S+', text)
print(emails)
```

Question:

What is the output of this code?

A) ['john.doe@example.com', 'jane_doe123@company.co.uk']

B) ['john.doe', 'jane_doe123']

C) ['example.com', 'company.co.uk']

D) An empty list

Answer: **A**

Explanation: The regular expression \S+@\S+ matches any non-whitespace characters before and after the "@" symbol. This extracts the full email addresses as shown.

37.

Python/Code Output

```
from nltk.stem import PorterStemmer
ps = PorterStemmer()
words = ["running", "ran", "runs", "easily", "fairly"]
```

stems = [ps.stem(word) for word in words]

```
print(stems)
```

Question:

What is the expected output of the above code?

A) ['run', 'ran', 'run', 'easili', 'fairli']

B) ['run', 'run', 'run', 'easili', 'fair']

C) ['running', 'ran', 'runs', 'easily', 'fairly']

D) ['run', 'ran', 'runs', 'easy', 'fair']

Answer: **A**

Explanation: PorterStemmer transforms "running" to "run", "ran" may remain "ran" (depending on the algorithm's rules), "runs" to "run", "easily" to "easili", and "fairly" to "fairli". Option A reflects these typical outputs.

38.

Python/Code Output

```
nums = [1, 2, 3, 4, 5, 6, 7, 8, 9]
result = list(filter(lambda x: x % 2 == 0, nums))
print(result)
```

Question:

What is printed by this code?

A) [1, 3, 5, 7, 9]

B) [2, 4, 6, 8]

C) [2, 4, 6, 8, 10]

D) [1, 2, 3, 4, 5, 6, 7, 8, 9]

Answer: **B**

Explanation: The lambda function filters out even numbers. The even numbers in the list are 2, 4, 6, and 8.

39.

Python/Code Output

```
def dummy_sentiment(text):
    if "good" in text:
        return "Positive"
    elif "bad" in text:
        return "Negative"
    else:
        return "Neutral"
```

```
print(dummy_sentiment("This movie is not bad"))
```

Question:

What does this function output for the given input?

A) Positive

B) Negative

C) Neutral

D) Error

Answer: B

Explanation: The function checks if the substring "good" is in the text; if not, it then checks for "bad". In "This movie is not bad", the substring "bad" is present, so the function returns "Negative" despite the presence of "not". This highlights a limitation in the simplistic approach.

40.

Python/Code Output

```
def flatten(lst):
    result = []
    for item in lst:
        if isinstance(item, list):
            result.extend(flatten(item))
        else:
            result.append(item)
    return result
```

```
print(flatten([1, [2, [3, 4], 5], 6]))
```

Question:

What is the output of the above code?

A) [1, [2, [3, 4], 5], 6]

B) [[1], [2], [3], [4], [5], [6]]

C) [1, 2, 3, 4, 5, 6]

D) An error occurs due to recursion depth.

Answer: C

Explanation: The recursive function “flatten” iterates through each item in the list; if an item is itself a list, it recurses and extends the result. The final flattened list is [1, 2, 3, 4, 5, 6].

Section E: Advanced AI, Computer Vision and Keras Case Studies

41.

Advanced Case Study

```
import pandas as pd
```

```
data = {
    'Region': ['North', 'North', 'South', 'South', 'East', 'East', 'West', 'West'],
    'Category': ['A', 'B', 'A', 'B', 'A', 'B', 'A', 'B'],
    'Sales': [100, 150, 200, 250, 300, 350, 400, 450]
}
df = pd.DataFrame(data)
grouped = df.groupby(['Region', 'Category']).agg({'Sales': ['sum', 'mean']})
print(grouped)
```

Question:

Which option best represents the MultiIndex DataFrame printed by this code?

A) Region and Category are the two index levels; each Region-Category pair has Sales sum and mean equal to its single Sales value.

B) Category and Region are the two index levels, with Category shown before Region.

C) Region and Category are shown, but the mean values are incorrectly half of the Sales values.

D) Only Region-level totals are printed; Category is not shown.

Answer: A

Explanation: The code groups the DataFrame by Region first and Category second. Since each Region-Category pair appears once, the sum and mean are the same as the original Sales value for that pair. The original file had duplicate-looking options for this question, so the options and answer have been corrected for clarity.

42.

Advanced Case Study

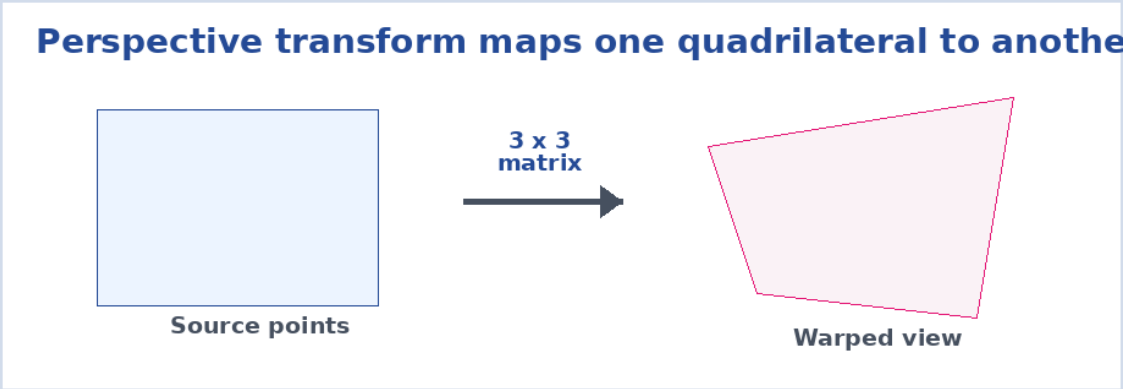
Consider this code snippet that applies a perspective transformation:

```
import cv2
import numpy as np
```

```
image = np.zeros((300, 300, 3), dtype=np.uint8)
pts1 = np.float32([[50, 50], [250, 50], [50, 250], [250, 250]])
pts2 = np.float32([[10, 100], [290, 50], [50, 290], [270, 300]])
M = cv2.getPerspectiveTransform(pts1, pts2)
warped = cv2.warpPerspective(image, M, (300, 300))
print(M.shape)
```

Question:

What is the shape of the transformation matrix M printed by the code?



A) (3, 3)

B) (4, 4)

C) (2, 3)

D) (3, 4)

Answer: **A**

Explanation: cv2.getPerspectiveTransform computes a 3×3 transformation matrix that maps four source points to four destination points. Thus, the printed shape is (3, 3).

43.

Advanced Case Study

Examine the following Keras Functional API code:

```
import tensorflow as tf
from tensorflow.keras.layers import Input, Conv2D, MaxPooling2D, Flatten, Dense, Concatenate
from tensorflow.keras.models import Model
```

```
input1 = Input(shape=(64, 64, 3))
```

x1 = Conv2D(16, (3, 3), activation='relu', padding='same')(input1)
x1 = MaxPooling2D((2, 2))(x1)

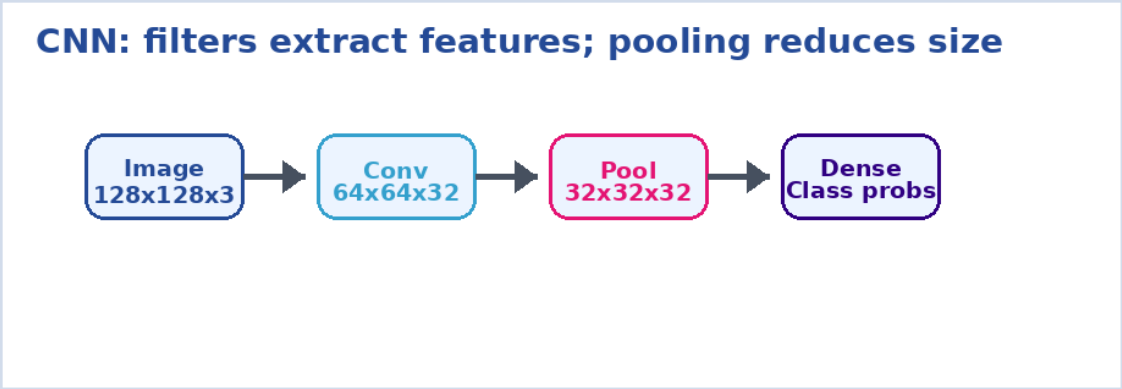
```
input2 = Input(shape=(64, 64, 3))
```

x2 = Conv2D(16, (5, 5), activation='relu', padding='same')(input2)
x2 = MaxPooling2D((2, 2))(x2)
merged = Concatenate()([x1, x2])
flat = Flatten()(merged)
output = Dense(10, activation='softmax')(flat)

```
model = Model(inputs=[input1, input2], outputs=output)
print(model.output_shape)
```

Question:

What is the shape of the model's output, excluding the batch size?



A) (10,)

B) (20,)

C) (None, 10)

D) (None, 20)

Answer: **A**

Explanation: The final Dense layer has 10 units with softmax activation, meaning that for each sample the network outputs a vector of shape (10,). Although the summary might show (None, 10) including the batch dimension, per-sample output shape is (10,).

44.

Advanced Case Study

A student writes a custom data generator for training a CNN. Consider this simplified code snippet:

```
import numpy as np
from tensorflow.keras.utils import Sequence
```

```
class CustomDataGenerator(Sequence):
    def __init__(self, data, labels, batch_size):
        self.data = data
        self.labels = labels
        self.batch_size = batch_size

    def __len__(self):
        return int(np.ceil(len(self.data) / self.batch_size))

    def __getitem__(self, idx):
        batch_x = self.data[idx * self.batch_size:(idx + 1) * self.batch_size]
        batch_y = self.labels[idx * self.batch_size:(idx + 1) * self.batch_size]
        # Normalize the batch data
        batch_x = batch_x / 255.0
        return batch_x, batch_y
```

```
data = np.random.randint(0, 256, (100, 32, 32, 3))
labels = np.random.randint(0, 10, 100)
gen = CustomDataGenerator(data, labels, 8)
x_batch, y_batch = gen.__getitem__(0)
print(x_batch.shape, y_batch.shape)
```

Question:

What output shape is printed by this code?

A) (8, 32, 32, 3) and (8,)

B) (8, 32, 32, 3) and (100,)

C) (100, 32, 32, 3) and (100,)

D) (12, 32, 32, 3) and (12,)

Answer: A

Explanation: The generator's batch size is 8. The first batch will have 8 samples of shape (32, 32, 3) and 8 corresponding labels. Thus, the shapes printed are (8, 32, 32, 3) for images and (8,) for labels.

45.

Advanced Case Study

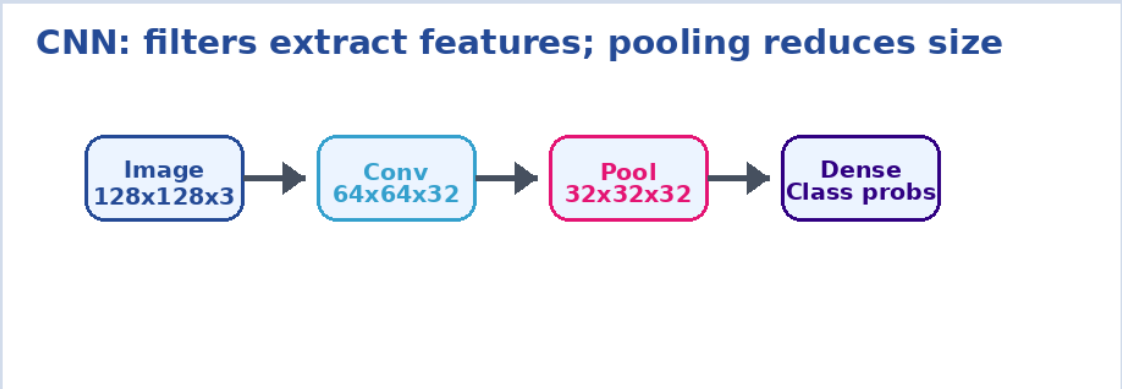
Consider the following CNN layer configuration:

```
import tensorflow as tf
from tensorflow.keras.layers import Input, Conv2D, MaxPooling2D
from tensorflow.keras.models import Model
```

```
inp = Input(shape=(128, 128, 3))
x = Conv2D(32, (3, 3), strides=(2, 2), padding='same')(inp)
x = MaxPooling2D(pool_size=(2, 2), padding='same')(x)
model = Model(inputs=inp, outputs=x)
print(model.output_shape)
```

Question:

What is the output shape of the model (excluding the batch dimension)?



A) (32, 32, 32)

B) (32, 32, 3)

C) (16, 16, 32)

D) (16, 16, 3)

Answer: **A**

Explanation: The Conv2D layer with stride 2 and padding "same" reduces spatial dimensions from 128×128 to 64×64 while outputting 32 channels. The MaxPooling2D with pool size 2 and padding "same" halves 64×64 to 32×32. Thus, the output shape is (32, 32, 32) if pooling halves each dimension. However, re-calculate carefully: After Conv2D: 128/2 = 64, so shape is (64, 64, 32). After MaxPooling2D: 64/2 = 32 (with padding same, sometimes remains 32), so shape is (32, 32, 32). Given options, option A (32, 32, 32) appears correct. Yet option C says (16, 16, 32) which would be if further halved. Let's check: MaxPooling2D with pool size (2,2) and stride default equal to pool size: if input is 64, output becomes ceil(64/2)=32. Thus the output is (32, 32, 32). So Answer: A (Corrected: Answer A)

46.

Advanced Case Study

Consider the following code using OpenCV to process an image:

```
import cv2
import numpy as np
```

```
image = cv2.imread('sample.png', 0)
kernel = cv2.getStructuringElement(cv2.MORPH_RECT, (5, 5))
opened = cv2.morphologyEx(image, cv2.MORPH_OPEN, kernel)
closed = cv2.morphologyEx(opened, cv2.MORPH_CLOSE, kernel)
print(closed.shape)
```

Question:

What is the effect of applying a morphological open followed by close, and what will be the shape of the output?

A) It removes noise and fills small holes; output shape is identical to the input.

B) It only removes noise; output shape is half the input size.

C) It blurs the image; output shape is larger than the input.

D) It detects edges; output shape is identical to the input.

Answer: A

Explanation: Morphological opening removes small noise, and closing fills small holes, while preserving the original image size. Thus, the output shape is the same as the input image's shape.

47.

Advanced Case Study

Consider the following code snippet:

```
from tensorflow.keras.applications import ResNet50
from tensorflow.keras.layers import GlobalAveragePooling2D, Dense
from tensorflow.keras.models import Model
```

```
base_model = ResNet50(weights='imagenet', include_top=False, input_shape=(224, 224, 3))
x = GlobalAveragePooling2D()(base_model.output)
predictions = Dense(5, activation='softmax')(x)
model = Model(inputs=base_model.input, outputs=predictions)
```

model.trainable = False

model.summary()

Question:

What does setting model.trainable = False achieve in this transfer learning context?

A) It freezes all layers of the model, preventing them from updating during training.

B) It automatically trains only the top Dense layer while fine-tuning the rest.

C) It unfreezes the base model for end-to-end training.

D) It resets the weights of the base model.

Answer: A

Explanation: Setting model.trainable = False freezes all layers in the model, meaning that during training, none of the weights (including the top Dense layer in this case) will be updated. Typically, one would freeze the base model and then unfreeze the top layers; here, the entire model is frozen.

48.

Advanced Case Study

A student needs to create a custom loss function to handle imbalanced class weights in a classification

problem. Consider the following code snippet:

```
import tensorflow as tf
```

```
def weighted_categorical_crossentropy(weights):
    def loss(y_true, y_pred):
        # Clip y_pred to prevent log(0)
        y_pred = tf.clip_by_value(y_pred, 1e-7, 1 - 1e-7)
        loss = -tf.reduce_sum(weights * y_true * tf.math.log(y_pred), axis=-1)
        return loss
    return loss
```

Assume weights = [1.0, 2.0, 1.0, 1.0, 3.0] for 5 classes.

```
loss_fn = weighted_categorical_crossentropy(tf.constant([1.0, 2.0, 1.0, 1.0, 3.0]))
```

Question:

What is the primary purpose of this custom loss function?

A) To reduce the overall training time.

B) To penalize misclassifications in minority classes more heavily.

C) To simplify the computation of categorical crossentropy.

D) To ensure that the loss is always positive.

Answer: **B**

Explanation: By multiplying the standard categorical crossentropy loss with class-specific weights, the loss function penalizes misclassification of classes with higher weights (typically minority classes) more severely, helping to mitigate class imbalance.

49.

Advanced Case Study

Consider this code using Keras' ImageDataGenerator:

```
from tensorflow.keras.preprocessing.image import ImageDataGenerator
import numpy as np
```

```
datagen = ImageDataGenerator(
    rotation_range=40,
    width_shift_range=0.2,
    height_shift_range=0.2,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True,
    fill_mode='nearest'
)
```

Dummy image: 256x256 RGB image

```
img = np.random.randint(0, 256, (256, 256, 3), dtype=np.uint8)
img = np.expand_dims(img, axis=0)
aug_iter = datagen.flow(img, batch_size=1)
```

augmented_img = next(aug_iter)

```
print(augmented_img.shape)
```

Question:

What is the shape of augmented_img?

A) (1, 256, 256, 3)

B) (256, 256, 3)

C) (1, 512, 512, 3)

D) (1, 128, 128, 3)

Answer: **A**

Explanation: The ImageDataGenerator preserves the image dimensions (256, 256, 3) and adds a batch dimension. Therefore, the output shape is (1, 256, 256, 3).

50.

Advanced Case Study

Consider the following advanced Keras callback:

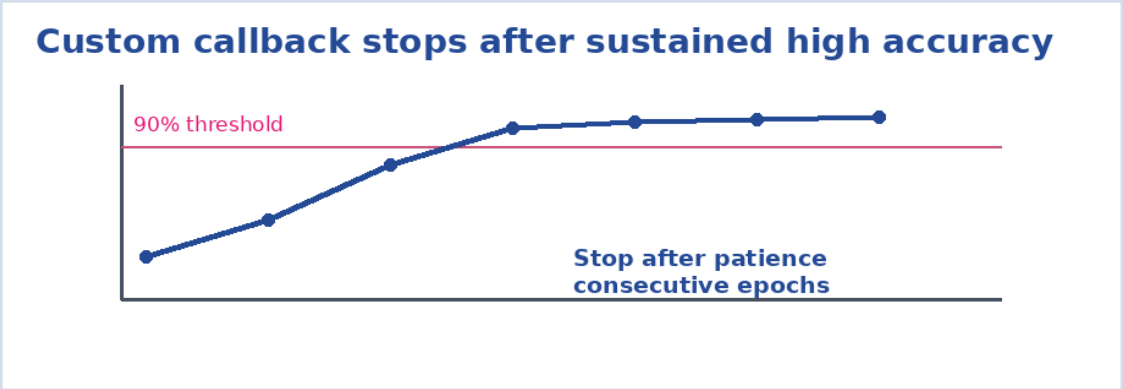
```
import tensorflow as tf
from tensorflow.keras.callbacks import Callback
```

```
class CustomEarlyStopping(Callback):
    def __init__(self, patience=3, threshold=0.90):
        super(CustomEarlyStopping, self).__init__()
        self.patience = patience
        self.threshold = threshold
        self.wait = 0
    def on_epoch_end(self, epoch, logs=None):
        current_acc = logs.get("accuracy")
        if current_acc is not None:
            if current_acc >= self.threshold:
                self.wait += 1
                if self.wait >= self.patience:
                    self.model.stop_training = True
            else:
                self.wait = 0
```

Example usage in model.fit(...)

Question:

What is the primary purpose of this custom callback?



A) To stop training after any single epoch reaches 90% accuracy.

B) To stop training if accuracy is at or above 90% for a specified number of consecutive epochs.

C) To dynamically adjust the learning rate when accuracy is low.

D) To save the model whenever accuracy exceeds 90%.

Answer: **B**

Explanation: This callback monitors the "accuracy" metric and increments a counter (wait) when accuracy is at or above the threshold (0.90). If this condition is met for patience consecutive epochs, training is stopped. This ensures training stops only after sustained high performance.

Answer Key

Q	Ans	Q	Ans	Q	Ans	Q	Ans	Q	Ans
1	B	2	B	3	B	4	B	5	C
6	B	7	B	8	C	9	B	10	C
11	A	12	B	13	C	14	B	15	A
16	A	17	C	18	C	19	B	20	C
21	B	22	B	23	A	24	A	25	A
26	B	27	B	28	B	29	B	30	B
31	A	32	A	33	B	34	C	35	A
36	A	37	A	38	B	39	B	40	C
41	A	42	A	43	A	44	A	45	A
46	A	47	A	48	B	49	A	50	B

Editorial Quality Note

This edition has been formatted for website/PDF publication with corrected instructions, consistent 50-question structure, clear question blocks, answer/explanation panels, and selected visual learning aids inside relevant question blocks.

SPACE FOR ROUGH WORK