Importing the dependencies and reading the data

```
!pip install contractions -q

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import itertools
import ast
import os
import re
from collections import defaultdict
import contractions

from wordcloud import WordCloud
import nltk
from nltk.tokenize import word_tokenize, sent_tokenize

from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import CountVectorizer,
TfidfVectorizer
from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import LabelEncoder
from sklearn.ensemble import RandomForestClassifier,
RandomForestRegressor
from sklearn.tree import DecisionTreeClassifier,
DecisionTreeRegressor, export_graphviz
from sklearn.base import BaseEstimator, ClassifierMixin
from sklearn.metrics import mean_absolute_error, f1_score,
precision_score, recall_score, classification_report, accuracy_score,
confusion_matrix
from sklearn.inspection import PartialDependenceDisplay

from xgboost import XGBClassifier


from sklearn.preprocessing import LabelEncoder, LabelBinarizer
from tensorflow.keras.utils import to_categorical
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences
from tensorflow.keras.backend import clear_session
from tensorflow.keras.models import Sequential, load_model
from tensorflow.keras.layers import Embedding, SpatialDropout1D,
Bidirectional, LSTM, Dense, Dropout, GlobalMaxPooling1D,
BatchNormalization, LeakyReLU
```

```python
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.regularizers import l2
from tensorflow.keras.utils import plot_model
from tensorflow.keras.callbacks import ReduceLROnPlateau,
ModelCheckpoint, EarlyStopping
from sklearn.utils.class_weight import compute_class_weight


nltk.download('punkt')
```

```
[nltk_data] Downloading package punkt to /usr/share/nltk_data...
[nltk_data]   Package punkt is already up-to-date!

True
```

```python
df =
pd.read_csv("/kaggle/input/multi-label-film-classifier/film_details.cs
v")
df.head()
```

```
                             Title Category  \
0                    Dekalog (1988)    movie
1                     The Godfather    movie
2  Lawrence of Arabia (re-release)    movie
3         The Leopard (re-release)    movie
4                   The Conformist    movie

                                                 Url  Metascore  \
0     https://www.metacritic.com/movie/dekalog-1988/        100
1    https://www.metacritic.com/movie/the-godfather/        100
2  https://www.metacritic.com/movie/lawrence-of-a...        100
3  https://www.metacritic.com/movie/the-leopard-r...        100
4  https://www.metacritic.com/movie/the-conformis...        100

   Number_of_critic_reviewers  User_score  Number_of_user_reviewers  \
0                          13         100                       112
1                          16         100                      4082
2                           8         100                       442
3                          12         100                        84
4                          11         100                       105

                                        Plot_summary  \
0  This masterwork by Krzysztof Kieślowski is one...
1  Francis Ford Coppola's epic features Marlon Br...
2  The 40th anniversary re-release of David Lean'...
3  Set in Sicily in 1860, Luchino Visconti's spec...
4  Set in Rome in the 1930s, this re-release of B...

                                       Genres
0                                   ['Drama']
1                          ['Crime', 'Drama']
```

```
2  ['Adventure', 'Biography', 'Drama', 'War']
3                        ['Drama', 'History']
4                                   ['Drama']
```

```python
# Keeping the relevant columns only

df = df[["Title", "Plot_summary", "Genres"]]
df.head()
```

```
                             Title  \
0                   Dekalog (1988)
1                    The Godfather
2  Lawrence of Arabia (re-release)
3         The Leopard (re-release)
4                   The Conformist

                                        Plot_summary  \
0  This masterwork by Krzysztof Kieślowski is one...
1  Francis Ford Coppola's epic features Marlon Br...
2  The 40th anniversary re-release of David Lean'...
3  Set in Sicily in 1860, Luchino Visconti's spec...
4  Set in Rome in the 1930s, this re-release of B...

                                       Genres
0                                   ['Drama']
1                          ['Crime', 'Drama']
2  ['Adventure', 'Biography', 'Drama', 'War']
3                        ['Drama', 'History']
4                                   ['Drama']
```

```python
# Check for missing values
missing_values = df.isnull().sum()
missing_values
```

```
Title           0
Plot_summary    0
Genres          0
dtype: int64
```

```python
# Check for completely empty rows
empty_rows = df[df.isnull().all(axis=1)]
print(f"\nNumber of completely empty rows: {len(empty_rows)}")
```

```
Number of completely empty rows: 0
```

Observation: No Null/Empty value exists in the dataset

Text preprocessing

```
def clean_text(text):
    if not isinstance(text, str):
        return ""

    text = text.lower()               # Lowercase the text
    text = re.sub(r'<.*?>', '', text) # Remove HTML tags (if any)
    text = re.sub(r'\s+', ' ', text)  # Replace non-breaking spaces
and special whitespace with regular space
    text = text.strip()               # Strip leading/trailing
whitespace
    return text


def remove_illegal_excel_chars(text):
    if isinstance(text, str):
        # Removes all control characters except for \t (tab), \n
(newline), and \r (carriage return)
        return re.sub(r'[\x00-\x08\x0B\x0C\x0E-\x1F\x7F]', '', text)
    return text


df['Plot_summary'] = df['Plot_summary'].apply(clean_text)
df

for col in df.select_dtypes(include='object').columns: # Apply to all
object (text) columns
    df[col] = df[col].apply(remove_illegal_excel_chars)
```

Exploratory Data Analysis (EDA)

```
# Word & Character Counts
df['word_count'] = df['Plot_summary'].apply(lambda x:
len(word_tokenize(x)))
df['char_count'] = df['Plot_summary'].apply(len)

# Sentence Count (for avg sentence length)
df['sentence_count'] = df['Plot_summary'].apply(lambda x:
len(sent_tokenize(x)))

# Vocabulary size (entire dataset)
all_words = [word.lower() for text in df['Plot_summary'] for word in
word_tokenize(text)]
vocab_size = len(set(all_words))
```

```python
print(f"Vocabulary Size: {vocab_size}")
```

```
Vocabulary Size: 43311
```

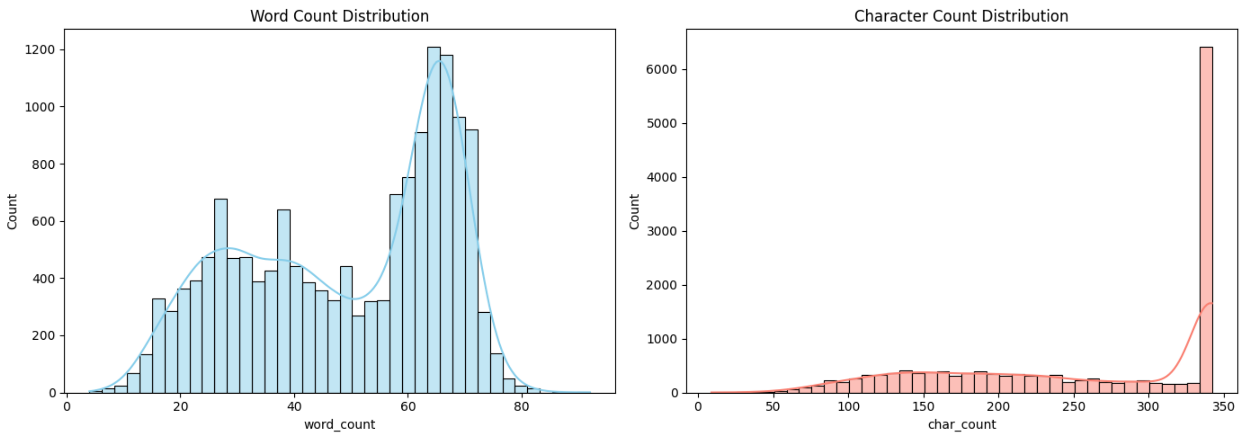
```python
df
```

```
                                    Title  \
0                           Dekalog (1988)
1                          The Godfather
2        Lawrence of Arabia (re-release)
3               The Leopard (re-release)
4                        The Conformist
...                                   ...
15149                             Cavemen
15150                             Work It
15151  Category 7: The End of the World
15152                             Stalker
15153                                Dads

                                           Plot_summary  \
0      this masterwork by krzysztof kieślowski is one...
1      francis ford coppola's epic features marlon br...
2      the 40th anniversary re-release of david lean'...
3      set in sicily in 1860, luchino visconti's spec...
4      set in rome in the 1930s, this re-release of b...
...                                                 ...
15149  cavemen revolves around joel, his younger brot...
15150  after they are laid off, lee standish (ben kol...
15151  "category 7: the end of the world" picks up wh...
15152  lt. beth davis (maggie q) leads the threat ass...
15153  the lives of video game company co-founders el...

                                                  Genres
word_count  \
0                                             ['Drama']          55

1                                    ['Crime', 'Drama']          60

2             ['Adventure', 'Biography', 'Drama', 'War']          25

3                                  ['Drama', 'History']          44

4                                             ['Drama']          43

...                                                 ...         ...

15149                              ['Comedy', 'Sci-Fi']          67

15150                                        ['Comedy']          35
```

```
15151  ['Action', 'Adventure', 'Drama', 'Sci-Fi', 'Th...          72

15152                         ['Crime', 'Drama', 'Thriller']          49

15153                                             ['Comedy']          36


       char_count  sentence_count
0             342               2
1             342               2
2             144               1
3             242               2
4             249               1
...           ...             ...
15149         342               4
15150         151               1
15151         340               3
15152         233               4
15153         184               1

[15154 rows x 6 columns]
```

```
plt.figure(figsize=(14, 5))

plt.subplot(1, 2, 1)
sns.histplot(df['word_count'], kde=True, bins=40, color='skyblue')
plt.title("Word Count Distribution")

plt.subplot(1, 2, 2)
sns.histplot(df['char_count'], kde=True, bins=40, color='salmon')
plt.title("Character Count Distribution")

plt.tight_layout()
plt.show()
```

```
/usr/local/lib/python3.10/dist-packages/seaborn/_oldcore.py:1119:
FutureWarning: use_inf_as_na option is deprecated and will be removed
in a future version. Convert inf values to NaN before operating
instead.
  with pd.option_context('mode.use_inf_as_na', True):
/usr/local/lib/python3.10/dist-packages/seaborn/_oldcore.py:1119:
FutureWarning: use_inf_as_na option is deprecated and will be removed
in a future version. Convert inf values to NaN before operating
instead.
  with pd.option_context('mode.use_inf_as_na', True):
```



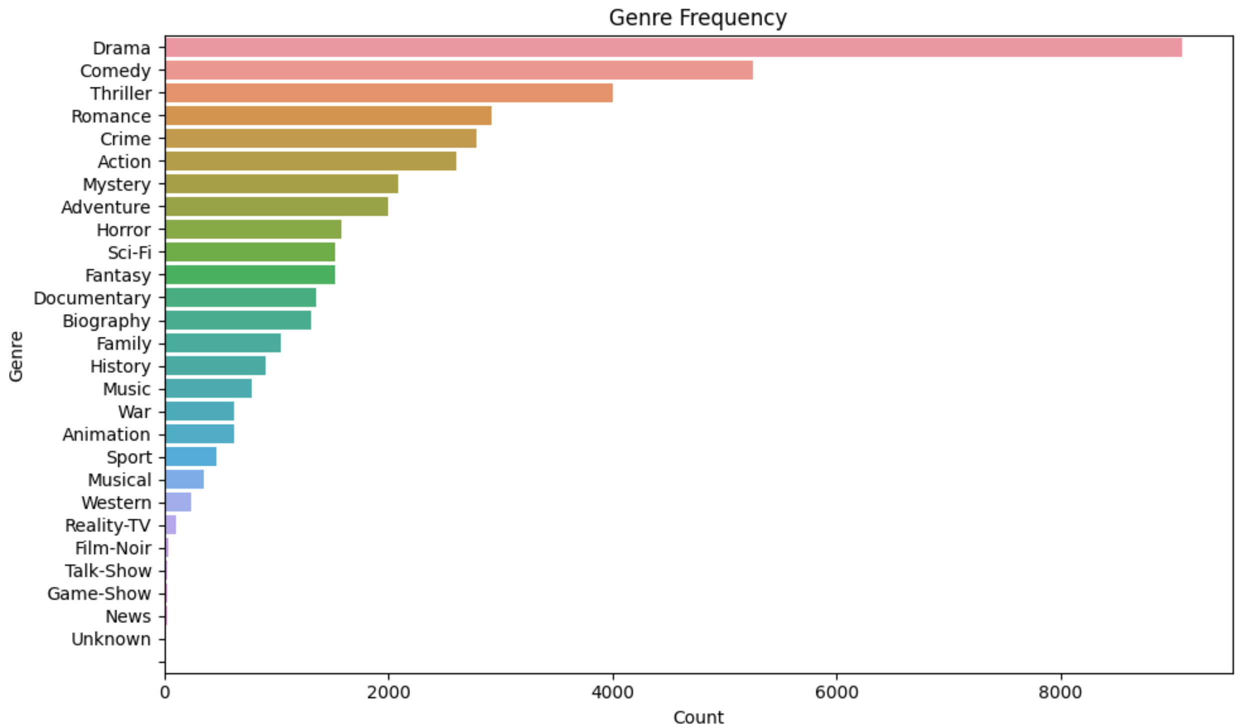
Observations:

1. There are two main plot length styles in the dataset — brief (30–40 words) and extended (60–70 words).
2. A maximum character limit of ~342 is enforced or commonly hit.
3. The dataset mixes single-sentence summaries with multi-sentence overviews.
4. Useful for training models where input size and richness vary, such as in summarization or classification tasks.

```
# Convert genre strings to lists
df['Genres'] = df['Genres'].apply(ast.literal_eval)

# Count number of unique genres
all_genres = list(itertools.chain.from_iterable(df['Genres']))
unique_genres = set(all_genres)
num_unique_genres = len(unique_genres)

print(num_unique_genres)
```

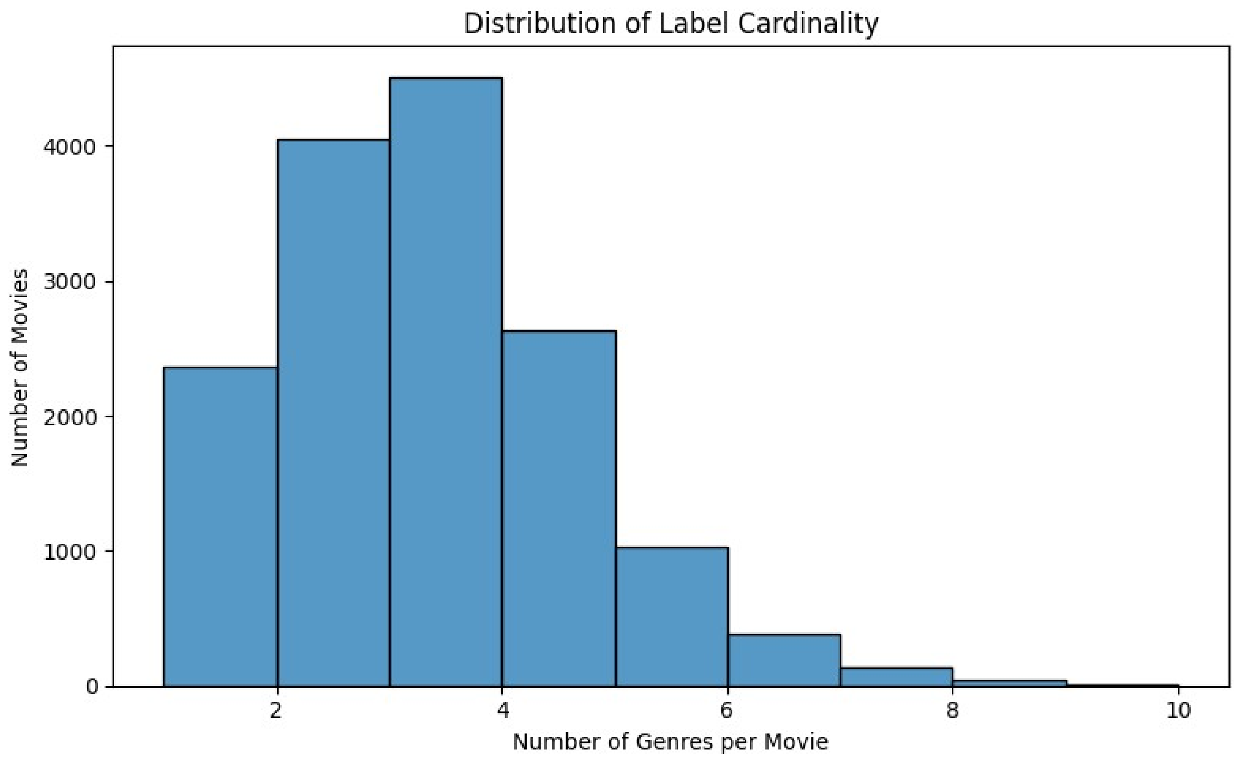
```
28
```

```
# Plot genre frequency
genre_counts = pd.Series(all_genres).value_counts()
plt.figure(figsize=(10, 6))
sns.barplot(x=genre_counts.values, y=genre_counts.index)
plt.title('Genre Frequency')
plt.xlabel('Count')
plt.ylabel('Genre')
plt.tight_layout()
plt.show()
```

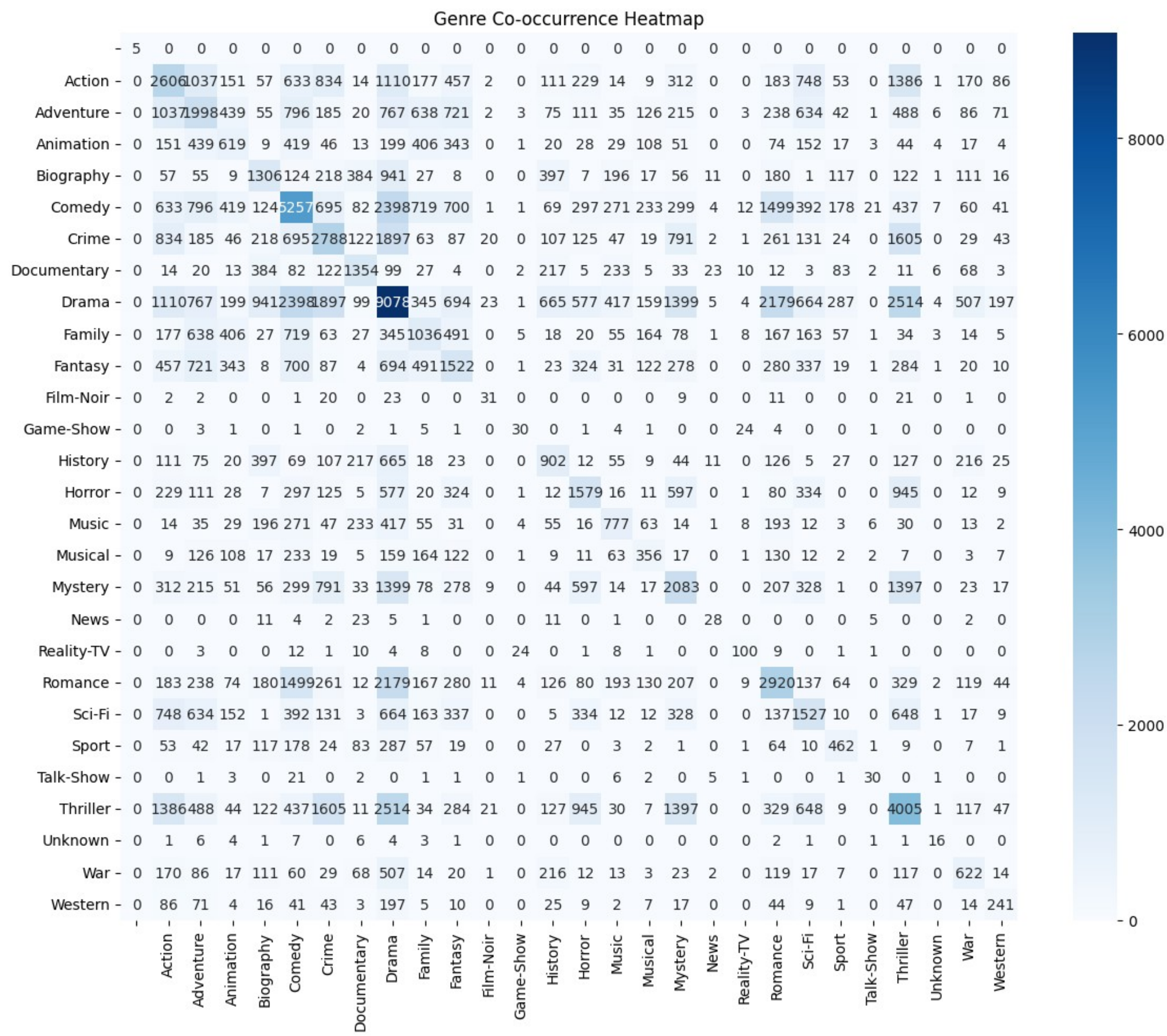



```python
# Distribution of label cardinality (genres per movie)
genre_counts_per_movie = df['Genres'].apply(len)
plt.figure(figsize=(8, 5))
sns.histplot(genre_counts_per_movie, bins=range(1,
genre_counts_per_movie.max()+2), kde=False)
plt.title('Distribution of Label Cardinality')
plt.xlabel('Number of Genres per Movie')
plt.ylabel('Number of Movies')
plt.tight_layout()
plt.show()
```

```
/usr/local/lib/python3.10/dist-packages/seaborn/_oldcore.py:1119:
FutureWarning: use_inf_as_na option is deprecated and will be removed
in a future version. Convert inf values to NaN before operating
instead.
  with pd.option_context('mode.use_inf_as_na', True):
```




```
# Visualize genre co-occurrence as heatmap

unique_genres_list = sorted(list(unique_genres)) # Convert set to
sorted list for DataFrame compatibility

co_occurrence = pd.DataFrame(0, index=unique_genres_list,
columns=unique_genres_list)
for genres in df['Genres']:
    for genre1, genre2 in
itertools.combinations_with_replacement(genres, 2):
        co_occurrence.loc[genre1, genre2] += 1
        if genre1 != genre2:
            co_occurrence.loc[genre2, genre1] += 1

plt.figure(figsize=(12, 10))
sns.heatmap(co_occurrence, annot=True, fmt="d", cmap="Blues")
plt.title('Genre Co-occurrence Heatmap')
plt.tight_layout()
plt.show()
```



Observations:

1. There are a total of 14 unique genres
2. Genre frequency: Drama Genre is most often
3. Label cardinality: Most of the movies have 3 genres
4. Genre co-occurrence: Drama is being classified as Drama very often

Word Cloud / Top N-Grams

```
# Combine all plot summaries
combined_text = " ".join(df['Plot_summary'].dropna())

# Generate overall word cloud
```

```
wordcloud = WordCloud(width=1000, height=500,
background_color='white').generate(combined_text)

plt.figure(figsize=(15, 7))
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis('off')
plt.title("Overall Word Cloud from Plot Summaries")
plt.show()
```


```
# Function to extract top n-grams
def get_top_ngrams(texts, ngram_range=(1, 1), top_n=20):
    vec = CountVectorizer(ngram_range=ngram_range,
stop_words='english')
    X = vec.fit_transform(texts)
    sum_words = X.sum(axis=0)
    words_freq = [(word, sum_words[0, idx]) for word, idx in
vec.vocabulary_.items()]
    return sorted(words_freq, key=lambda x: x[1], reverse=True)
[:top_n]

# Extract top unigrams, bigrams, trigrams
top_unigrams = get_top_ngrams(df['Plot_summary'], (1, 1))
top_bigrams = get_top_ngrams(df['Plot_summary'], (2, 2))
top_trigrams = get_top_ngrams(df['Plot_summary'], (3, 3))

# Combine into a DataFrame for display
top_ngrams_df = pd.DataFrame({
    'Unigrams': [u[0] for u in top_unigrams],
    'Unigram_Freq': [u[1] for u in top_unigrams],
```

```python
    'Bigrams': [b[0] for b in top_bigrams],
    'Bigram_Freq': [b[1] for b in top_bigrams],
    'Trigrams': [t[0] for t in top_trigrams],
    'Trigram_Freq': [t[1] for t in top_trigrams],
})

top_ngrams_df
```

```
   Unigrams  Unigram_Freq         Bigrams  Bigram_Freq  \
0      life          2450        year old          797
1       new          2185        new york          612
2     young          1716     high school          355
3     world          1621       york city          275
4    family          1593     los angeles          260
5     story          1485      true story          240
6    series          1216      small town          219
7       man          1196     young woman          202
8       old          1185     best friend          202
9      year          1172     tells story          183
10     film          1131    series based          168
11     love          1086       world war          155
12    years          1036    best friends          155
13      set           937   sony pictures          149
14    based           934       young man          147
15    lives           845     warner bros          146
16   comedy           831    20th century          142
17     home           829          war ii          107
18    woman           827     century fox          104
19     time           821  series created          104

                    Trigrams  Trigram_Freq
0              new york city           275
1               world war ii           106
2           20th century fox            87
3     sony pictures classics            77
4           based true story            67
5            new line cinema            59
6               year old son            55
7                12 year old            52
8   premiered originally uk            49
9                17 year old            48
10             year old girl            45
11         year old daughter            44
12          lions gate films            43
13               16 year old            43
14               14 year old            40
15               15 year old            37
16      limited series based            34
17          coming age story            29
```

```
18              11 year old          28
19            year old boy           27
```

Observations:

1. Common words: *life*, *new*, *young*, *world*, *family*, *man*, *love* focuses on **personal journeys**, **relationships**, and **family dynamics** and frequent themes of **youth** and **self-discovery**
2. Top phrases: *year old*, *new york*, *high school*, *los angeles* emphasises on **age/life stage**, **urban settings**, and **school themes**
3. Common trigrams are *new york city*, *world war ii*, *based true story* and plots include **historical events**, **real locations**, and **biographical elements** which mentions of **production companies** suggest some summaries mix marketing content
4. High-frequency verbs/nouns: *find*, *discover*, *follow*, *story*, *family* indicates themes of **transformation**, **search**, and **human-centered stories**
5. Strong presence of **drama**, **family**, **coming-of-age**, **romance**, **action** settings often in **big cities** or **historic periods**

```
df.to_excel("film_df.xlsx",index  = False)
```

Data preparation

```
df = pd.read_excel("/kaggle/working/film_df.xlsx")
df
```

```
                                   Title  \
0                          Dekalog (1988)
1                          The Godfather
2        Lawrence of Arabia (re-release)
3               The Leopard (re-release)
4                          The Conformist
...                                   ...
15149                             Cavemen
15150                             Work It
15151  Category 7: The End of the World
15152                             Stalker
15153                                Dads

                                            Plot_summary  \
0      this masterwork by krzysztof kieślowski is one...
1      francis ford coppola's epic features marlon br...
2      the 40th anniversary re-release of david lean'...
3      set in sicily in 1860, luchino visconti's spec...
4      set in rome in the 1930s, this re-release of b...
...                                                  ...
15149  cavemen revolves around joel, his younger brot...
15150  after they are laid off, lee standish (ben kol...
```

```
15151  "category 7: the end of the world" picks up wh...
15152  lt. beth davis (maggie q) leads the threat ass...
15153  the lives of video game company co-founders el...

                                                 Genres  
word_count  \
0                                             ['Drama']           55

1                                    ['Crime', 'Drama']           60

2              ['Adventure', 'Biography', 'Drama', 'War']           25

3                                  ['Drama', 'History']           44

4                                             ['Drama']           43

...                                                 ...          ...

15149                              ['Comedy', 'Sci-Fi']           67

15150                                        ['Comedy']           35

15151  ['Action', 'Adventure', 'Drama', 'Sci-Fi', 'Th...           72

15152                     ['Crime', 'Drama', 'Thriller']           49

15153                                        ['Comedy']           36


       char_count  sentence_count
0             342               2
1             342               2
2             144               1
3             242               2
4             249               1
...           ...             ...
15149         342               4
15150         151               1
15151         340               3
15152         233               4
15153         184               1

[15154 rows x 6 columns]

# Making single genre from multi genre (Same plot_summary and Title
will be present)

df['Genres'] = df['Genres'].apply(ast.literal_eval)
df = df.explode('Genres')

# Remove rows with blank or null genres
```

```
df = df[df['Genres'].str.strip() != '']
df = df[df['Genres'].notnull()]

df = df.drop_duplicates(subset='Plot_summary', keep='first')
df
```

```
                                   Title  \
0                          Dekalog (1988)
1                          The Godfather
2       Lawrence of Arabia (re-release)
3              The Leopard (re-release)
4                        The Conformist
...                                  ...
15149                            Cavemen
15150                            Work It
15151  Category 7: The End of the World
15152                            Stalker
15153                               Dads

                                            Plot_summary     Genres  \
0      this masterwork by krzysztof kieślowski is one...      Drama
1      francis ford coppola's epic features marlon br...      Crime
2      the 40th anniversary re-release of david lean'...  Adventure
3      set in sicily in 1860, luchino visconti's spec...      Drama
4      set in rome in the 1930s, this re-release of b...      Drama
...                                                  ...        ...
15149  cavemen revolves around joel, his younger brot...     Comedy
15150  after they are laid off, lee standish (ben kol...     Comedy
15151  "category 7: the end of the world" picks up wh...     Action
15152  lt. beth davis (maggie q) leads the threat ass...      Crime
15153  the lives of video game company co-founders el...     Comedy

       word_count  char_count  sentence_count
0              55         342               2
1              60         342               2
2              25         144               1
3              44         242               2
4              43         249               1
...           ...         ...             ...
15149          67         342               4
15150          35         151               1
15151          72         340               3
15152          49         233               4
15153          36         184               1

[15086 rows x 6 columns]
```

```
np.unique(df["Genres"])
```

```
array(['Action', 'Adventure', 'Animation', 'Biography', 'Comedy',
'Crime',
       'Documentary', 'Drama', 'Family', 'Fantasy', 'Film-Noir',
       'Game-Show', 'History', 'Horror', 'Music', 'Musical',
'Mystery',
       'News', 'Reality-TV', 'Romance', 'Sci-Fi', 'Sport', 'Talk-
Show',
       'Thriller', 'Unknown', 'War', 'Western'], dtype=object)
```

```
# Group plot summaries by genre
grouped = df.groupby('Genres')['Plot_summary'].apply(lambda texts: '
'.join(texts))

print(grouped)
```

```
Genres
Action          seven samurai (shichinin no samurai) tells the...
Adventure       the 40th anniversary re-release of david lean'...
Animation       a living puppet, with the help of a cricket as...
Biography       in 1431, jeanne d'arc is placed on trial on ch...
Comedy          a silent film production company and cast make...
Crime           francis ford coppola's epic features marlon br...
Documentary     two inner-city chicago boys with hopes of beco...
Drama           this masterwork by krzysztof kieślowski is one...
Family          set in the gloriously vibrant town of cobbleto...
Fantasy         henry spencer tries to survive his industrial ...
Film-Noir       pulp novelist holly martins travels to shadowy...
Game-Show       hosted by alan cummings, 20 contestants (inclu...
History         12 mighty orphans tells the true story of the ...
Horror          a phoenix secretary embezzles $40,000 from her...
Music           spike lee’s adaptation of the broadway show “p...
Musical         in the hilltops of burundi, a group of escaped...
Mystery         a wheelchair-bound photographer spies on his n...
News            the morning talk show hosted by former fox new...
Reality-TV      the sundance reality show takes a look at the ...
Romance         lily bart (anderson) is a ravishing socialite ...
Sci-Fi          want out of your life? just pay the fee and we...
Sport           former espn commentator bill simmons hosts a n...
Talk-Show       david letterman returns to the talk show world...
Thriller        a serial murderer is strangling women with a n...
Unknown         a woman watches time passing next to the suitc...
War             an epic romantic drama about two czech pilots,...
Western         notorious gunfighter jimmy ringo rides into to...
Name: Plot_summary, dtype: object
```

```
vectorizer = TfidfVectorizer(stop_words='english', max_features=1000)
# Making a TF-IDF vectorizer
tfidf_matrix = vectorizer.fit_transform(grouped)

tfidf_df = pd.DataFrame(tfidf_matrix.toarray(), index=grouped.index,
```

```
columns=vectorizer.get_feature_names_out()) # A df with TF-IDF values

top_words_per_genre = {} # Top 10 words per genre
for genre in tfidf_df.index:
    top_indices = np.argsort(tfidf_df.loc[genre])[::-1][:10]
    top_words = [(tfidf_df.columns[i], tfidf_df.loc[genre,
tfidf_df.columns[i]]) for i in top_indices]
    top_words_per_genre[genre] = top_words

print(top_words_per_genre)
```

```
{'Action': [('world', 0.19943485663134644), ('new',
0.1958628890498895), ('life', 0.17494304512161366), ('war',
0.15272048812443206), ('series', 0.13894532390032455), ('man',
0.13872826496836338), ('years', 0.1380200303105427), ('agent',
0.13248268466990923), ('young', 0.13102113913678762), ('action',
0.12205194215772648)], 'Adventure': [('new', 0.2080679696729915),
('young', 0.1851306932306889), ('life', 0.18044044547532517),
('adventure', 0.1719643706202739), ('family', 0.17120199256242452),
('world', 0.16945741859965285), ('story', 0.15217954894437735),
('father', 0.1426809631495456), ('time', 0.13884801992301668),
('year', 0.12766191439696184)], 'Animation': [('animated',
0.3966200164277816), ('world', 0.23223993154650027), ('new',
0.2013955656379807), ('voiced', 0.18789663064105505), ('life',
0.1770456700963849), ('young', 0.16833728626381556), ('family',
0.167797140563768), ('adventure', 0.16050327005929182), ('comedy',
0.15493334640606762), ('story', 0.14021404896424447)], 'Biography':
[('story', 0.43152818039074436), ('life', 0.3178481846088339),
('true', 0.27943218238417056), ('based', 0.18789055247812841),
('world', 0.16751967095348902), ('young', 0.1566292433155204), ('war',
0.1402934750572954), ('film', 0.13032635044696775), ('man',
0.128014285402108), ('family', 0.10788204509768609)], 'Comedy':
[('comedy', 0.33112446807380574), ('life', 0.2707947185268614),
('new', 0.2192950911581916), ('family', 0.1896656527931573), ('love',
0.15290246912512157), ('friends', 0.15182156714194048), ('old',
0.13697725821248444), ('best', 0.1360126611364208), ('year',
0.13328625424867496), ('school', 0.13196845958621659)], 'Crime':
[('drama', 0.20869356038031217), ('crime', 0.20448668198571338),
('life', 0.20040070266649943), ('police', 0.19205856643695396),
('detective', 0.18376168043310728), ('series', 0.1768030209120499),
('murder', 0.17547167981206485), ('young', 0.16683250012831588),
('new', 0.15986574954492064), ('family', 0.14561008420775928)],
'Documentary': [('documentary', 0.4716325094595509), ('film',
0.26386287329054925), ('life', 0.23029673831433498), ('world',
0.19452147727115046), ('years', 0.17587452734193185), ('story',
0.17482138645964482), ('new', 0.13550000339828003), ('footage',
0.1327961724740042), ('interviews', 0.12913234210333924), ('year',
0.12223059571547502)], 'Drama': [('life', 0.26087318388778075),
('family', 0.2322579733676066), ('young', 0.22855021642357207),
('story', 0.20222865963926956), ('new', 0.189997561852), ('old',
```

```
0.16596616878666745), ('year', 0.14954243333382014), ('love',
0.1488325020682016), ('mother', 0.12663564441274364), ('world',
0.12518357876408578)], 'Family': [('reality', 0.25298010853663955),
('deal', 0.2194785597916421), ('business', 0.19976270943794222),
('small', 0.18309319548227762), ('dream', 0.18309319548227762),
('game', 0.17563245878373113), ('based', 0.1620975930241725), ('town',
0.1620975930241725), ('american', 0.15006989857108785), ('cinema',
0.1154360065923957)], 'Fantasy': [('young', 0.2858326129206947 3),
('house', 0.1751725909954738), ('old', 0.17391500054443745), ('world',
0.16559104674483946), ('new', 0.1490319420703555), ('mysterious',
0.1468485497622955), ('horror', 0.13732989491395653), ('island',
0.13732989491395653), ('supernatural', 0.13226371344301782),
('family', 0.125870185510539)], 'Film-Noir': [('man',
0.35129339135929866), ('woman', 0.3381202563397295), ('memory',
0.24725193927772082), ('harry', 0.24725193927772082), ('accused',
0.22504119468795583), ('prove', 0.22504119468795583), ('prison',
0.22504119468795583), ('try', 0.22504119468795583), ('murder',
0.2152902460471471), ('works', 0.2152902460471471)], 'Game-Show':
[('reality', 0.5524175665497283), ('series', 0.3945839761069488),
('win', 0.2235584463646521), ('challenges', 0.14681451081902366),
('million', 0.1402103275263119), ('chance', 0.13413506781879128),
('features', 0.13413506781879128), ('men', 0.12327368125715428),
('named', 0.12327368125715428), ('women', 0.11837519283208464)],
'History': [('football', 0.5175147982518294), ('playing',
0.42837905862654085), ('course', 0.23443631467922044), ('state',
0.22389062417511252), ('texas', 0.22389062417511252), ('spirit',
0.22389062417511252), ('tells', 0.22389062417511252), ('winning',
0.21418952931327043), ('great', 0.21418952931327043), ('12',
0.20520771836786567)], 'Horror': [('family', 0.2710175236590921),
('young', 0.2019415105918361), ('night', 0.17358532433590018), ('new',
0.172599001459398), ('home', 0.16155338390028365), ('life',
0.16086852282514535), ('friends', 0.15090748476472177), ('town',
0.14723559716199744), ('horror', 0.13776559516567907), ('house',
0.13501080781681127)], 'Music': [('american', 0.32822891434581253),
('music', 0.3003424471669747), ('film', 0.27665570233171477),
('young', 0.2557625126728605 4), ('dance', 0.2089921630081137),
('winning', 0.2089921630081137), ('host', 0.20022829811131648),
('man', 0.17050834178190702), ('portrait', 0.13320368731348306),
('intimate', 0.12623929076053161)], 'Musical': [('jane',
0.28007209976472425), ('wild', 0.28007209976472425), ('west',
0.2451538135201308), ('school', 0.23516421205787819), ('day',
0.22581956396261182), ('cold', 0.1545637846295604), ('brought',
0.14003604988236212), ('lady', 0.14003604988236212), ('star',
0.14003604988236212), ('female', 0.14003604988236212)], 'Mystery':
[('life', 0.2053386190310571), ('murder', 0.2017388654093303),
('young', 0.19750879986388464), ('sony', 0.15462332788909386),
('discovers', 0.14242883674128906), ('boyfriend',
0.13901324360757317), ('assigned', 0.1324852329217956), ('share',
0.1324852329217956), ('killer', 0.12652562608925508), ('new',
```

```
0.12205069188929618)], 'News': [('news', 0.7340266246503894),
('interviews', 0.47409885348645), ('features', 0.35111076628717547),
('fox', 0.3363873083581224), ('film', 0.0), ('films', 0.0), ('final',
0.0), ('finally', 0.0), ('finding', 0.0), ('finds', 0.0)], 'Reality-
TV': [('reality', 0.5172203564822967), ('series', 0.3940726525579404),
('lives', 0.1970363262789702), ('new', 0.18995080674937304),
('feature', 0.14586316176768518), ('10', 0.13369138047003942),
('family', 0.13369138047003942), ('challenges', 0.12218688377242458),
('living', 0.10259494705216218), ('real', 0.10259494705216218)],
'Romance': [('having', 0.21364923743616013), ('husband',
0.21364923743616013), ('professor', 0.19519770077944507), ('romantic',
0.19519770077944507), ('trying', 0.1870122920388786), ('mother',
0.17226341959555275), ('life', 0.1655672802344535), ('time',
0.1592539921138186), ('young', 0.1592539921138186), ('love',
0.15328213385481979)], 'Sci-Fi': [('space', 0.3640289443946534),
('earth', 0.26607204899287246), ('life', 0.22568311676956526),
('eddie', 0.21841736663679207), ('girlfriend', 0.19955403674465436),
('set', 0.16280815228413986), ('frank', 0.14561157775786138),
('alive', 0.1390615062173222), ('scientist', 0.1390615062173222),
('control', 0.1390615062173222)], 'Sport': [('pop',
0.5874327616786038), ('culture', 0.532218874648506), ('cover',
0.48625448838240987), ('new', 0.3677259747454447), ('filmmaker', 0.0),
('films', 0.0), ('final', 0.0), ('finally', 0.0), ('finding', 0.0),
('finds', 0.0)], 'Talk-Show': [('returns', 0.5157221268968978),
('president', 0.33556971233457705), ('chicago', 0.31981148231926015),
('george', 0.30542534544104794), ('david', 0.29219138238318865),
('television', 0.29219138238318865), ('season', 0.27993864638411065),
('series', 0.2578610634484489), ('live', 0.24783761435690044),
('world', 0.2209673400784671)], 'Thriller': [('family',
0.2585767369693155), ('town', 0.22892531076518188), ('leader',
0.1771215072853537), ('recently', 0.1688039465822609), ('london',
0.1612106085835627), ('thriller', 0.15422541475346627), ('small',
0.1477581354110374), ('son', 0.14173724238731697), ('life',
0.13081446329438964), ('man', 0.125826343685526)], 'Unknown':
[('comedy', 0.2751370307830225), ('short', 0.26276049271626817),
('stories', 0.2513751813608843), ('henry', 0.2513751813608843),
('story', 0.24083402950071067), ('businessman', 0.1518402611516397),
('dog', 0.14434699717923205), ('bob', 0.14434699717923205), ('tom',
0.13756851539151124), ('master', 0.13756851539151124)], 'War':
[('air', 0.3458906221264l836), ('epic', 0.3458906221264l836),
('drama', 0.3296477259911001), ('fall', 0.3296477259911001), ('ii',
0.3148191236116359), ('force', 0.3148191236116359), ('romantic',
0.30117813174906566), ('war', 0.30117813174906566), ('love',
0.23650497173154297), ('woman', 0.23650497173154297)], 'Western':
[('trouble', 0.35975994341462764), ('finds', 0.2900711410431522),
('town', 0.27879563757515896), ('young', 0.26816481012969), ('guy',
0.18874328689773856), ('notorious', 0.18874328689773856), ('captain',
0.17987997170731382), ('local', 0.17178839889740752), ('attack',
0.17178839889740752), ('turns', 0.17178839889740752)]}
```

```python
# Print the result
for genre, words in top_words_per_genre.items():
    print(f"\n{genre}:\n" + ", ".join([f"{word} ({score:.3f})" for
word, score in words]))
```

```
Action:
world (0.199), new (0.196), life (0.175), war (0.153), series (0.139),
man (0.139), years (0.138), agent (0.132), young (0.131), action
(0.122)

Adventure:
new (0.208), young (0.185), life (0.180), adventure (0.172), family
(0.171), world (0.169), story (0.152), father (0.143), time (0.139),
year (0.128)

Animation:
animated (0.397), world (0.232), new (0.201), voiced (0.188), life
(0.177), young (0.168), family (0.168), adventure (0.161), comedy
(0.155), story (0.140)

Biography:
story (0.432), life (0.318), true (0.279), based (0.188), world
(0.168), young (0.157), war (0.140), film (0.130), man (0.128), family
(0.108)

Comedy:
comedy (0.331), life (0.271), new (0.219), family (0.190), love
(0.153), friends (0.152), old (0.137), best (0.136), year (0.133),
school (0.132)

Crime:
drama (0.209), crime (0.204), life (0.200), police (0.192), detective
(0.184), series (0.177), murder (0.175), young (0.167), new (0.160),
family (0.146)

Documentary:
documentary (0.472), film (0.264), life (0.230), world (0.195), years
(0.176), story (0.175), new (0.136), footage (0.133), interviews
(0.129), year (0.122)

Drama:
life (0.261), family (0.232), young (0.229), story (0.202), new
(0.190), old (0.166), year (0.150), love (0.149), mother (0.127),
world (0.125)

Family:
reality (0.253), deal (0.219), business (0.200), small (0.183), dream
(0.183), game (0.176), based (0.162), town (0.162), american (0.150),
cinema (0.115)
```

```
Fantasy:
young (0.286), house (0.175), old (0.174), world (0.166), new (0.149),
mysterious (0.147), horror (0.137), island (0.137), supernatural
(0.132), family (0.126)

Film-Noir:
man (0.351), woman (0.338), memory (0.247), harry (0.247), accused
(0.225), prove (0.225), prison (0.225), try (0.225), murder (0.215),
works (0.215)

Game-Show:
reality (0.552), series (0.395), win (0.224), challenges (0.147),
million (0.140), chance (0.134), features (0.134), men (0.123), named
(0.123), women (0.118)

History:
football (0.518), playing (0.428), course (0.234), state (0.224),
texas (0.224), spirit (0.224), tells (0.224), winning (0.214), great
(0.214), 12 (0.205)

Horror:
family (0.271), young (0.202), night (0.174), new (0.173), home
(0.162), life (0.161), friends (0.151), town (0.147), horror (0.138),
house (0.135)

Music:
american (0.328), music (0.300), film (0.277), young (0.256), dance
(0.209), winning (0.209), host (0.200), man (0.171), portrait (0.133),
intimate (0.126)

Musical:
jane (0.280), wild (0.280), west (0.245), school (0.235), day (0.226),
cold (0.155), brought (0.140), lady (0.140), star (0.140), female
(0.140)

Mystery:
life (0.205), murder (0.202), young (0.198), sony (0.155), discovers
(0.142), boyfriend (0.139), assigned (0.132), share (0.132), killer
(0.127), new (0.122)

News:
news (0.734), interviews (0.474), features (0.351), fox (0.336), film
(0.000), films (0.000), final (0.000), finally (0.000), finding
(0.000), finds (0.000)

Reality-TV:
reality (0.517), series (0.394), lives (0.197), new (0.190), feature
(0.146), 10 (0.134), family (0.134), challenges (0.122), living
(0.103), real (0.103)
```

```
Romance:
having (0.214), husband (0.214), professor (0.195), romantic (0.195),
trying (0.187), mother (0.172), life (0.166), time (0.159), young
(0.159), love (0.153)

Sci-Fi:
space (0.364), earth (0.266), life (0.226), eddie (0.218), girlfriend
(0.200), set (0.163), frank (0.146), alive (0.139), scientist (0.139),
control (0.139)

Sport:
pop (0.587), culture (0.532), cover (0.486), new (0.368), filmmaker
(0.000), films (0.000), final (0.000), finally (0.000), finding
(0.000), finds (0.000)

Talk-Show:
returns (0.516), president (0.336), chicago (0.320), george (0.305),
david (0.292), television (0.292), season (0.280), series (0.258),
live (0.248), world (0.221)

Thriller:
family (0.259), town (0.229), leader (0.177), recently (0.169), london
(0.161), thriller (0.154), small (0.148), son (0.142), life (0.131),
man (0.126)

Unknown:
comedy (0.275), short (0.263), stories (0.251), henry (0.251), story
(0.241), businessman (0.152), dog (0.144), bob (0.144), tom (0.138),
master (0.138)

War:
air (0.346), epic (0.346), drama (0.330), fall (0.330), ii (0.315),
force (0.315), romantic (0.301), war (0.301), love (0.237), woman
(0.237)

Western:
trouble (0.360), finds (0.290), town (0.279), young (0.268), guy
(0.189), notorious (0.189), captain (0.180), local (0.172), attack
(0.172), turns (0.172)
```

Observations:

1. "Life", "New", "Young", and "World" appear frequently across multiple genres — indicating common thematic foundations in storytelling.
2. Genre-specific keywords highlight narrative focus:
   - War: “war”, “ii”, “soldiers”, “army” — strong historical/military emphasis.
   - Romance: “love”, “woman”, “man”, “family” — emotionally driven relationships.

- Crime & Thriller: “murder”, “detective”, “police”, “mysterious” — classic crime elements.
- Sci-Fi: “earth”, “future”, “time” — futuristic and speculative themes.
- Documentary: “documentary”, “film”, “footage”, “interviews” — indicative of real-world storytelling.

3. Character-focused genres:
   - Biography: “true”, “based”, “years”, “man” — emphasizing factual recounts.
   - Animation & Family: “adventure”, “voiced”, “boy”, “girl” — often geared toward younger audiences.
4. Entertainment & Format-driven genres:
   - Game-Show / Reality-TV / Talk-Show: “reality”, “series”, “win”, “daily”, “live” — format-specific vocabulary.
   - Musical / Music: “music”, “band”, “rock”, “musical” — creative performance language.
5. Emotional tone distinction:
   - Comedy: “comedy”, “friends”, “school”, “old” — lighthearted and nostalgic.
   - Horror: “town”, “home”, “night”, “mysterious” — eerie, unsettling settings.
6. Unique standout terms:
   - Western: “sheriff”, “texas”, “town” — regionally specific storytelling.
   - Film-Noir: “woman”, “prove”, “private”, “husband” — classic noir dynamics.

Enlisting important words as features from the plot_summary column

```
# Fit TF-IDF vectorizer on all plot summaries
vectorizer = TfidfVectorizer(stop_words='english', max_features=1000)
tfidf_matrix = vectorizer.fit_transform(df['Plot_summary'])

# Map index to words
feature_names = np.array(vectorizer.get_feature_names_out())

# Extract top N words per row
def extract_top_keywords(row_index, top_n=5):
    row = tfidf_matrix[row_index].toarray().flatten()
    top_indices = row.argsort()[::-1][:top_n]
    return feature_names[top_indices].tolist()

# Apply to all rows and store in a new column
df['important_words'] = [extract_top_keywords(i, top_n=5) for i in 
range(tfidf_matrix.shape[0])]

# Join into a string to make a readable column
df['important_words'] = df['important_words'].apply(lambda words: ', 
'.join(words))
df
```

```
                                 Title  \
0                        Dekalog (1988)
1                        The Godfather
2       Lawrence of Arabia (re-release)
3              The Leopard (re-release)
4                        The Conformist
...                                 ...
15149                           Cavemen
15150                           Work It
15151  Category 7: The End of the World
15152                           Stalker
15153                              Dads

                                            Plot_summary     Genres  \
0      this masterwork by krzysztof kieślowski is one...      Drama
1      francis ford coppola's epic features marlon br...      Crime
2      the 40th anniversary re-release of david lean'...  Adventure
3      set in sicily in 1860, luchino visconti's spec...      Drama
4      set in rome in the 1930s, this re-release of b...      Drama
...                                                  ...        ...
15149  cavemen revolves around joel, his younger brot...     Comedy
15150  after they are laid off, lee standish (ben kol...     Comedy
15151  "category 7: the end of the world" picks up wh...     Action
15152  lt. beth davis (maggie q) leads the threat ass...      Crime
15153  the lives of video game company co-founders el...     Comedy

       word_count  char_count  sentence_count  \
0              55         342               2
1              60         342               2
2              25         144               1
3              44         242               2
4              43         249               1
...           ...         ...             ...
15149          67         342               4
15150          35         151               1
15151          72         340               3
15152          49         233               4
15153          36         184               1

                                          important_words
0       complex, emotional, person, greatest, originally
1                    family, oscar, role, near, portrait
2                      peter, david, history, film, food
3      ancient, greatest, cinema, international, adap...
4                  louis, jean, professor, feature, sent
...                                                  ...
15149                   andy, pilot, service, nick, kate
15150                          lee, ben, women, men, new
15151            world, rest, nation, threatens, chicago
15152              davis, recent, unit, threat, includes
```

```
15153                 green, upside, video, peter, martin

[15086 rows x 7 columns]
```

```
df.to_excel("final_df.xlsx",index = False)
```

Modeling

```
#Keeing the relevant columns only

final_df = pd.read_excel("/kaggle/working/final_df.xlsx")

# Display the first few rows and check the relevant columns
final_df[['Plot_summary', 'important_words', 'Genres']].head()
```

```
                                        Plot_summary  \
0  this masterwork by krzysztof kieślowski is one...
1  francis ford coppola's epic features marlon br...
2  the 40th anniversary re-release of david lean'...
3  set in sicily in 1860, luchino visconti's spec...
4  set in rome in the 1930s, this re-release of b...

                                     important_words     Genres
0   complex, emotional, person, greatest, originally      Drama
1                family, oscar, role, near, portrait      Crime
2                  peter, david, history, film, food  Adventure
3  ancient, greatest, cinema, international, adap...      Drama
4               louis, jean, professor, feature, sent      Drama
```

```
# Combine Plot Summary and Important Words
final_df['combined_text'] = final_df['Plot_summary'] + ' ' +
final_df['important_words']

# TF-IDF Vectorization
vectorizer = TfidfVectorizer(stop_words='english', max_features=5000)
X = vectorizer.fit_transform(final_df['combined_text'])

# Encode Genres
label_encoder = LabelEncoder()
y = label_encoder.fit_transform(final_df['Genres'])

# Train-Test Split
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)
```

Logisic Regression

```
# Train Logistic Regression with fast solver
lr_model = LogisticRegression(max_iter=200, solver='saga')
lr_model.fit(X_train, y_train)
lr_preds = lr_model.predict(X_test)

lr_accuracy = accuracy_score(y_test, lr_preds)
lr_accuracy
```

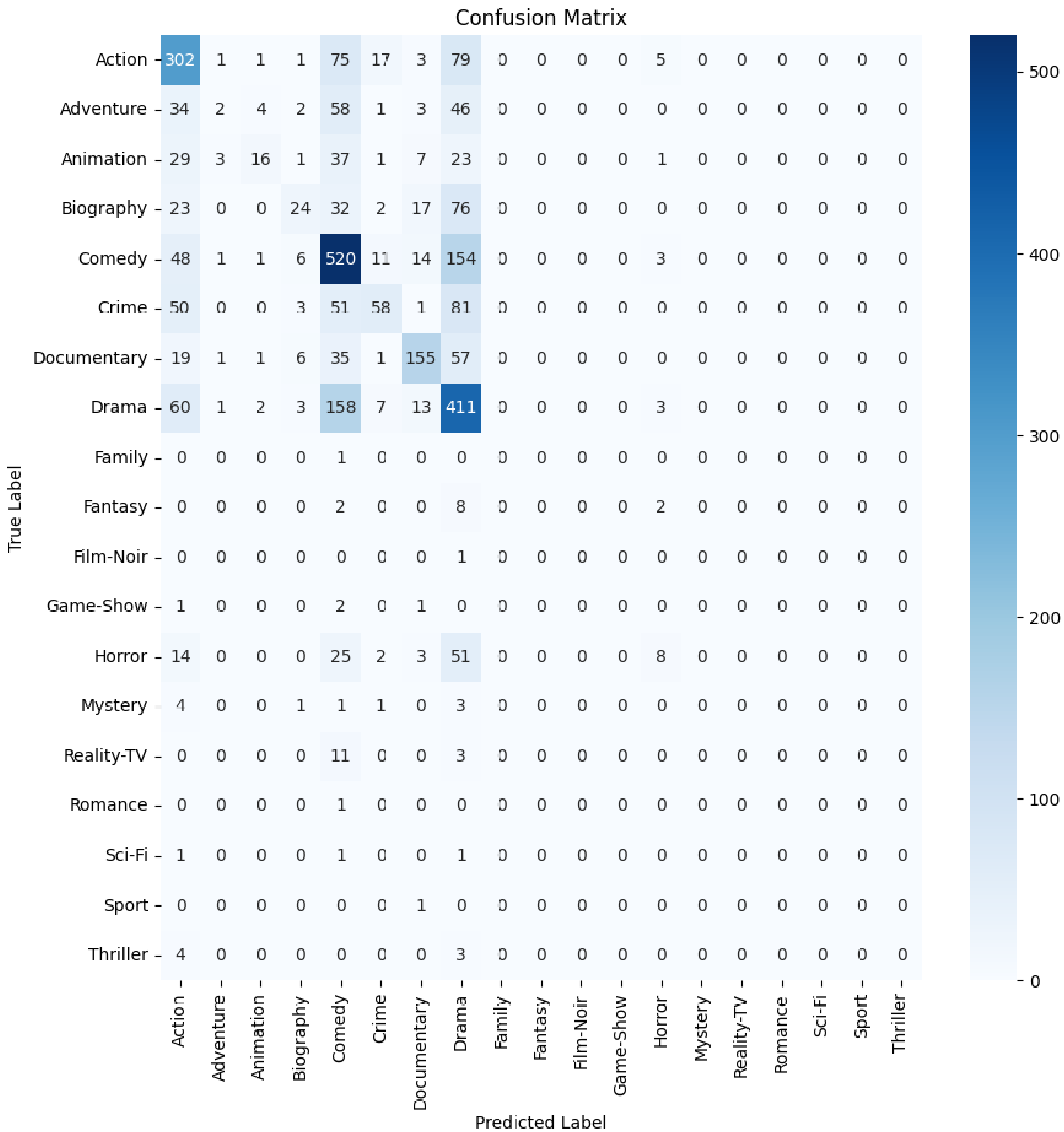
```
0.49569251159708416
```

```
unique_labels = np.unique(y_test)
target_names=label_encoder.inverse_transform(unique_labels)
lr_report = classification_report(
    y_test,
    lr_preds,
    labels=unique_labels,
    target_names=target_names,
    output_dict=True
)

for k, v in lr_report.items():
    print(f"{k} : {v}")
```

```
Action : {'precision': 0.5127334465195246, 'recall':
0.6239669421487604, 'f1-score': 0.5629077353215284, 'support': 484}
Adventure : {'precision': 0.2222222222222222, 'recall':
0.013333333333333334, 'f1-score': 0.025157232704402517, 'support':
150}
Animation : {'precision': 0.64, 'recall': 0.13559322033898305, 'f1-
score': 0.2237762237762238, 'support': 118}
Biography : {'precision': 0.5106382978723404, 'recall':
0.13793103448275862, 'f1-score': 0.21719457013574664, 'support': 174}
Comedy : {'precision': 0.5148514851485149, 'recall':
0.6860158311345647, 'f1-score': 0.5882352941176472, 'support': 758}
Crime : {'precision': 0.5742574257425742, 'recall':
0.23770491803278687, 'f1-score': 0.33623188405797094, 'support': 244}
Documentary : {'precision': 0.7110091743119266, 'recall':
0.5636363636363636, 'f1-score': 0.6288032454361054, 'support': 275}
Drama : {'precision': 0.41223671013039115, 'recall':
0.6246200607902735, 'f1-score': 0.49667673716012084, 'support': 658}
Family : {'precision': 0.0, 'recall': 0.0, 'f1-score': 0.0, 'support':
1}
Fantasy : {'precision': 0.0, 'recall': 0.0, 'f1-score': 0.0,
'support': 12}
Film-Noir : {'precision': 0.0, 'recall': 0.0, 'f1-score': 0.0,
'support': 1}
Game-Show : {'precision': 0.0, 'recall': 0.0, 'f1-score': 0.0,
'support': 4}
Horror : {'precision': 0.36363636363636365, 'recall':
```

```
0.07766990291262135, 'f1-score': 0.128, 'support': 103}
Mystery : {'precision': 0.0, 'recall': 0.0, 'f1-score': 0.0,
'support': 10}
Reality-TV : {'precision': 0.0, 'recall': 0.0, 'f1-score': 0.0,
'support': 14}
Romance : {'precision': 0.0, 'recall': 0.0, 'f1-score': 0.0,
'support': 1}
Sci-Fi : {'precision': 0.0, 'recall': 0.0, 'f1-score': 0.0, 'support':
3}
Sport : {'precision': 0.0, 'recall': 0.0, 'f1-score': 0.0, 'support':
1}
Thriller : {'precision': 0.0, 'recall': 0.0, 'f1-score': 0.0,
'support': 7}
accuracy : 0.49569251159708416
macro avg : {'precision': 0.23482026976757145, 'recall':
0.16318271614791816, 'f1-score': 0.1687885748794603, 'support': 3018}
weighted avg : {'precision': 0.4905491870637692, 'recall':
0.49569251159708416, 'f1-score': 0.457673789465906, 'support': 3018}

/usr/local/lib/python3.10/dist-packages/sklearn/metrics/
_classification.py:1344: UndefinedMetricWarning: Precision and F-score
are ill-defined and being set to 0.0 in labels with no predicted
samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classificatio
n.py:1344: UndefinedMetricWarning: Precision and F-score are ill-
defined and being set to 0.0 in labels with no predicted samples. Use
`zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classificatio
n.py:1344: UndefinedMetricWarning: Precision and F-score are ill-
defined and being set to 0.0 in labels with no predicted samples. Use
`zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

```
cm = confusion_matrix(y_test, lr_preds) # Confusion matrix
# cm = cm.astype("float") / cm.sum(axis=1)[:, np.newaxis]
plt.figure(figsize=(10, 10))
sns.heatmap(cm, annot=True, cmap="Blues", fmt="d",
xticklabels=target_names, yticklabels=target_names)
plt.title("Confusion Matrix")
plt.ylabel("True Label")
plt.xlabel("Predicted Label")
plt.savefig('lr_cf.png')
plt.show()
```



Random Forest Classifier

```
# Train Random Forest
rf_model = RandomForestClassifier(n_estimators=100, random_state=42)
rf_model.fit(X_train, y_train)
rf_preds = rf_model.predict(X_test)
```

```python
rf_accuracy = accuracy_score(y_test, rf_preds)
rf_accuracy
```

```
0.4562624254473161
```

```python
rf_report = classification_report(
    y_test,
    rf_preds,
    labels=unique_labels,
    target_names=label_encoder.inverse_transform(unique_labels),
    output_dict=True
)

for k, v in rf_report.items():
    print(f"{k} : {v}")
```

```
Action : {'precision': 0.47424511545293074, 'recall':
0.5516528925619835, 'f1-score': 0.5100286532951289, 'support': 484}
Adventure : {'precision': 0.0, 'recall': 0.0, 'f1-score': 0.0,
'support': 150}
Animation : {'precision': 0.75, 'recall': 0.1271186440677966, 'f1-
score': 0.21739130434782608, 'support': 118}
Biography : {'precision': 0.47368421052631576, 'recall':
0.15517241379310345, 'f1-score': 0.23376623376623376, 'support': 174}
Comedy : {'precision': 0.4692526017029328, 'recall':
0.6543535620052771, 'f1-score': 0.5465564738292011, 'support': 758}
Crime : {'precision': 0.4818181818181818, 'recall':
0.21721311475409835, 'f1-score': 0.2994350282485876, 'support': 244}
Documentary : {'precision': 0.6884422110552764, 'recall':
0.49818181818181817, 'f1-score': 0.5780590717299577, 'support': 275}
Drama : {'precision': 0.378, 'recall': 0.574468085106383, 'f1-score':
0.4559710494571773, 'support': 658}
Family : {'precision': 0.0, 'recall': 0.0, 'f1-score': 0.0, 'support':
1}
Fantasy : {'precision': 0.0, 'recall': 0.0, 'f1-score': 0.0,
'support': 12}
Film-Noir : {'precision': 0.0, 'recall': 0.0, 'f1-score': 0.0,
'support': 1}
Game-Show : {'precision': 0.0, 'recall': 0.0, 'f1-score': 0.0,
'support': 4}
Horror : {'precision': 0.375, 'recall': 0.02912621359223301, 'f1-
score': 0.05405405405405406, 'support': 103}
Mystery : {'precision': 0.0, 'recall': 0.0, 'f1-score': 0.0,
'support': 10}
Reality-TV : {'precision': 1.0, 'recall': 0.07142857142857142, 'f1-
score': 0.13333333333333333, 'support': 14}
Romance : {'precision': 0.0, 'recall': 0.0, 'f1-score': 0.0,
'support': 1}
Sci-Fi : {'precision': 0.0, 'recall': 0.0, 'f1-score': 0.0, 'support':
3}
```

```
Sport : {'precision': 0.0, 'recall': 0.0, 'f1-score': 0.0, 'support':
1}
Thriller : {'precision': 0.0, 'recall': 0.0, 'f1-score': 0.0,
'support': 7}
accuracy : 0.4562624254473161
macro avg : {'precision': 0.26791801687134936, 'recall':
0.15151133239427708, 'f1-score': 0.15939974747692107, 'support': 3018}
weighted avg : {'precision': 0.4520819764762949, 'recall':
0.4562624254473161, 'f1-score': 0.41980210415546576, 'support': 3018}

/usr/local/lib/python3.10/dist-packages/sklearn/metrics/
_classification.py:1344: UndefinedMetricWarning: Precision and F-score
are ill-defined and being set to 0.0 in labels with no predicted
samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classificatio
n.py:1344: UndefinedMetricWarning: Precision and F-score are ill-
defined and being set to 0.0 in labels with no predicted samples. Use
`zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classificatio
n.py:1344: UndefinedMetricWarning: Precision and F-score are ill-
defined and being set to 0.0 in labels with no predicted samples. Use
`zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

```
unique_labels = np.unique(y_test)
target_names=label_encoder.inverse_transform(unique_labels)

# Plotting the confusion matrix
cm = confusion_matrix(y_test, rf_preds) # Confusion matrix
# cm = cm.astype("float") / cm.sum(axis=1)[:, np.newaxis]
plt.figure(figsize=(10, 10))
sns.heatmap(cm, annot=True, cmap="Blues", fmt="d",
xticklabels=target_names, yticklabels=target_names)
plt.title("Confusion Matrix")
plt.ylabel("True Label")
plt.xlabel("Predicted Label")
plt.savefig('rf_cf.png')
plt.show()
```

Custom Deep Learning Model

```
#Print the numbers of each class

print(df['Genres'].value_counts())

print()
```

```python
#create pie chart
numbers = df['Genres'].value_counts()
labels=df['Genres'].value_counts().index
```

```
Genres
Comedy          3820
Drama           3456
Action          2426
Documentary     1345
Crime           1194
Biography        823
Adventure        631
Animation        603
Horror           524
Fantasy           63
Reality-TV        48
Thriller          34
Mystery           33
Game-Show         20
Sci-Fi            14
Romance           11
Family            10
Unknown            6
Musical            6
Music              5
Film-Noir          4
Western            4
Talk-Show          2
History            1
War                1
Sport              1
News               1
Name: count, dtype: int64
```

```python
#Dropping less frequent rows (fewer than 10 entries)

# Count genre frequencies
genre_counts = final_df['Genres'].value_counts()

# Filter out genres with less than 10 entries
valid_genres = genre_counts[genre_counts >= 10].index
df =
final_df[final_df['Genres'].isin(valid_genres)].reset_index(drop=True)

# Display trimmed value counts
df['Genres'].value_counts()
```

```
Genres
Comedy          3820
```

```
Drama             3456
Action            2426
Documentary       1345
Crime             1194
Biography          823
Adventure          631
Animation          603
Horror             524
Fantasy             63
Reality-TV          48
Thriller            34
Mystery             33
Game-Show           20
Sci-Fi              14
Romance             11
Family              10
Name: count, dtype: int64

X = df['Plot_summary']
y = df['Genres']

X_train, X_temp, y_train, y_temp = train_test_split(X, y,
test_size=0.2, stratify=y, random_state=42)
X_val, X_test, y_val, y_test = train_test_split(X_temp, y_temp,
test_size=0.5, stratify=y_temp, random_state=42)

label_encoder = LabelEncoder() # Initialize the LabelEncoder

# Both learns the mapping and transforms the y_train labels
accordingly.
y_train = label_encoder.fit_transform(y_train)

# Applies the same mapping learned from the train train split to other
splits to ensure consistent encoding
y_val = label_encoder.transform(y_val)
y_test = label_encoder.transform(y_test)

# Converting the integer-encoded labels into one-hot encoded format
y_train = to_categorical(y_train)
y_val = to_categorical(y_val)
y_test = to_categorical(y_test)

#Cleaning text further

def preprocess_text(text):
    text = text.lower() #ensures that the text is uniform in case
    text = re.sub(r'(covid[-_]?19|covid2019|covid[-_]?2019|corona[-_]?
virus|corona|covid)', 'covid', text) #normalization to reduce variance
in texts/terms
    text = re.sub(r'http\S+', '', text) #eliminate web links from the
text
```

```
    text = re.sub(r'@\w+', '', text) # removes any social media
handles
    text = re.sub(r'#', '', text) #removes any hashtags from texts
    text = re.sub(r'\n', ' ', text) #removes any new lines from the
texts
    text = re.sub(r'\t', ' ', text) #replaces any tab characters with
a space
    text = re.sub(r'\r', ' ', text) #replaces any carriage return
characters with a space
    text = re.sub(r'â|â’', "'", text) #replaces any specific
characters appearing due to encoding issues with an apostrophe
    text = re.sub(r'\x92|\xa0|\x85|\x95', '', text) #removes various
unwanted characters appearing due to encoding artifacts
    text = contractions.fix(text) # expands shortened words using the
contractions library
    text = re.sub(r'[^\w\s]', ' ', text) # removes all characters that
are not word characters or whitespace
    return text #returns the cleaned and processed text

X_train = np.array([preprocess_text(text) for text in X_train])
X_val = np.array([preprocess_text(text) for text in X_val])
X_test = np.array([preprocess_text(text) for text in X_test])

# Tokenizing the texts

tokenizer = Tokenizer(filters='')
tokenizer.fit_on_texts(X_train)
word_counts = len(tokenizer.word_index) + 1  # vocabulary size
print("Numbers of unique words present in the TRAIN split:",
word_counts)

# print()

# tokenizer_test  = Tokenizer(filters='')
# tokenizer_test.fit_on_texts(X_val)
# word_counts_test = len(tokenizer_test.word_index) + 1  # vocabulary
size
# print("Numbers of unique words present in the TEST split:",
word_counts_test)

# print()

# tokenizer_val = Tokenizer(filters='')
# tokenizer_val.fit_on_texts(X_val)
# word_counts_val = len(tokenizer_val.word_index) + 1  # vocabulary
size
# print("Numbers of unique words present in the VALID split:",
word_counts_val)

Numbers of unique words present in the TRAIN split: 34610
```

```
#Vectorizing the text


train_sequences = tokenizer.texts_to_sequences(X_train)

maxlen = max([len(seq) for seq in train_sequences])
print("Maximum length of all sequences:", maxlen)

Maximum length of all sequences: 73

# Padding the sequences (Post-padding the sequences)

padded_train_sequences = pad_sequences(train_sequences, maxlen=maxlen,
padding='post')
print("Padded TRAINING Sequences Shape:",
padded_train_sequences.shape)

Padded TRAINING Sequences Shape: (12044, 73)

padded_train_sequences

array([[    1,   171,   155, ...,     0,     0,     0],
       [  171,    68,  2923, ...,     0,     0,     0],
       [    2, 19177,  2808, ...,     0,     0,     0],
       ...,
       [10233,   419,     1, ...,     0,     0,     0],
       [    1,  1903,   118, ...,     0,     0,     0],
       [10083,  6736,  6150, ...,     0,     0,     0]], dtype=int32)



longest_sequence_index = np.argmax([len(seq) for seq in
train_sequences])

# Get the longest sequence and its corresponding original sentence
longest_sequence = train_sequences[longest_sequence_index]
longest_sentence = X_train[longest_sequence_index]

print(f"Longest sequence index: \n{longest_sequence_index}")
print()
print(f"Longest sequence: \n{longest_sequence}")
print()
print(f"Longest sequence length: \n{len(longest_sequence)}")
print()
print(f"Longest sentence: \n{longest_sentence}")


Longest sequence index:
1905

Longest sequence:
[5126, 7, 2, 5900, 767, 20, 30, 173, 233, 1, 2196, 70, 3, 1529, 1480,
```

```
6, 8, 23, 22, 17, 169, 17, 343, 56, 55, 154, 5, 164, 17, 306, 5, 1375,
80, 1, 328, 3, 1, 219, 5, 7073, 60, 17, 7, 3987, 17, 54, 49, 1260, 5,
1627, 12, 8, 90, 2052, 28, 396, 1, 122, 17, 169, 21, 128, 102, 123,
204, 683, 23, 102, 55, 1388, 5, 164, 36]

Longest sequence length:
73

Longest sentence:
rex is a cab driver who has never left the mining town of broken hill
in his life  when he discovers he does not have long to live  he
decides to drive through the heart of the country to darwin  where he
is heard he will be able to die on his own terms  but along the way he
discovers that before you can end your life you have got to live it
```

```
# For Validation set
val_sequences = tokenizer.texts_to_sequences(X_val)
padded_val_sequences = pad_sequences(val_sequences, maxlen=maxlen,
padding='post')

print("Padded VALIDATION Sequences Shape:",
padded_val_sequences.shape)
```

```
Padded VALIDATION Sequences Shape: (1505, 73)
```

```
# For test set
test_sequences = tokenizer.texts_to_sequences(X_test)
padded_test_sequences = pad_sequences(test_sequences, maxlen=maxlen,
padding='post')

print("Padded TEST Sequences Shape:", padded_test_sequences.shape)
```

```
Padded TEST Sequences Shape: (1506, 73)
```

```
# Creating the Embedding matrix using GloVe embedding

def create_embedding_matrix(filepath, word_index, embedding_dim):
    vocab_size = len(word_index) + 1  # Adding again 1 because of
reserved 0 index
    embedding_matrix = np.zeros((vocab_size, embedding_dim))

    with open(filepath) as f:
        for line in f:
            word, *vector = line.split()
            if word in word_index:
                idx = word_index[word]
                embedding_matrix[idx] = np.array(
                    vector, dtype=np.float32)[:embedding_dim]

    return embedding_matrix
```

```
embedding_dim = 100
filepath = '/kaggle/input/glove6b100dtxt/glove.6B.100d.txt'
embedding_matrix = create_embedding_matrix(filepath,
tokenizer.word_index, embedding_dim)

nonzero_elements = np.count_nonzero(np.count_nonzero(embedding_matrix,
axis=1))
print(f"Percent of vocabulary covered:
{round(nonzero_elements/word_counts*100, 2)}%")
```

```
Percent of vocabulary covered: 92.21%
```

```
# Get the words that are not covered by GloVe
not_covered_words = []
for word, idx in tokenizer.word_index.items():
    if np.count_nonzero(embedding_matrix[idx]) == 0:  # If the
embedding vector is all zeros
        not_covered_words.append(word)

# Print some of the words that are not covered
print(f"Total uncovered words: {len(not_covered_words)}")
print()
print("Sample of uncovered words:", not_covered_words[:50])
```

```
Total uncovered words: 2695

Sample of uncovered words: ['acirc', 'covid', 'jaeden', 'haddish',
'britbox', 'vikander', 'kumail', 'nanjiani', 'stre', 'throu',
'docuseries', 'roiland', 'daveed', 'lakeith', 'mulaney', 'parvana',
'reynor', 'negga', 'exarchopoulos', 'awkwafina', 'mahershala',
'thomasin', 'boutella', 'rosow', 'minhee', 'schoenaerts', '64257',
'nélisse', 'horri', 'dangero', 'krieps', 'demián', 'caestecker',
'vanderham', 'boyega', 'qualley', 'hirut', 'impos', 'polaha',
'ansiedad', 'efira', 'americ', 'erivo', 'astrof', 'documentry',
'ménochet', 'delevingne', 'konkle', 'maslany', 'raffey']
```

The Dataset is ready to be fed to the neural network upto this point.

```
Train features = padded_train_sequences

Train target = y_train

Validation features = padded_val_sequences

Validation target = y_val
```

```
Test features = padded_test_sequences

Test target = y_test

print(padded_train_sequences)  # training feature
print(y_train)  # training target

print()
print("=========================================================")
print()

print(len(padded_train_sequences))  # training feature length
print(len(y_train))  # training target length

[[    1   171   155 ...     0     0     0]
 [  171    68  2923 ...     0     0     0]
 [    2 19177  2808 ...     0     0     0]
 ...
 [10233   419     1 ...     0     0     0]
 [    1  1903   118 ...     0     0     0]
 [10083  6736  6150 ...     0     0     0]]
[[0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]
 ...
 [0. 0. 0. ... 0. 0. 0.]
 [1. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]]

=========================================================

12044
12044

# Creating the model



clear_session ()

model = Sequential()

model.add(Embedding(word_counts,                   # using the pre-
trained embedding matrix for word embeddings
                    embedding_dim,                           # convert
each word in a sequence to a dense vector of size embedding_dim
                    weights=[embedding_matrix],
                    input_length=maxlen,
                    trainable=True))

model.add(SpatialDropout1D(0.3)) # dropout to the embedding layer to
```

```
prevent overfitting (randomly drops entire feature maps rather than
individual elements)

model.add(Bidirectional(LSTM(units=64, return_sequences=True))) #
bidirectional LSTM layer has 64 units and outputs sequence
model.add(BatchNormalization()) # stabilizing and accelerating the
training by normalizing each layer’s input
model.add(Dropout(0.25))

model.add(Bidirectional(LSTM(units=32, return_sequences=False))) #
bidirectional LSTM layer has 32 units and outputs sequence
model.add(BatchNormalization())
model.add(Dropout(0.2))

model.add(Dense(24, activation='relu', kernel_regularizer=l2(0.05))) #
This dense layer consisting of 24 neurons with ReLU activation
functions process the LSTM outputs
model.add(BatchNormalization())
model.add(Dropout(0.3))

model.add(Dense(16, activation='relu', kernel_regularizer=l2(0.05))) #
This dense layer consisting of 16 neurons with ReLU activation
functions process the LSTM outputs
model.add(BatchNormalization())
model.add(Dropout(0.2))

model.add(Dense(17, activation='softmax')) # output layer of 5 neurons
for 5 classes ; softmax activation to output the class with maximum
probability
```

```
/usr/local/lib/python3.10/dist-packages/keras/src/layers/core/
embedding.py:90: UserWarning: Argument `input_length` is deprecated.
Just remove it.
  warnings.warn(
```

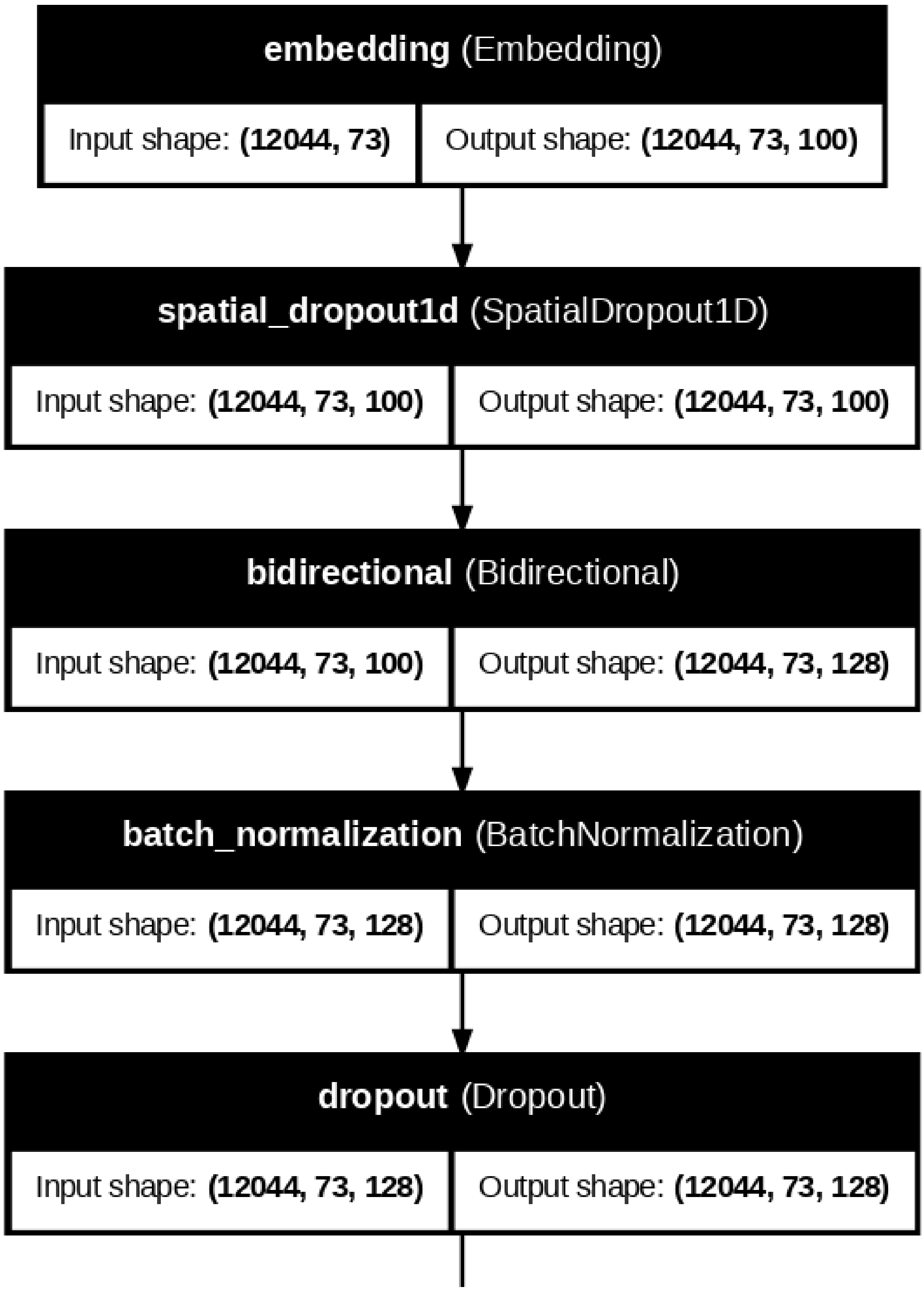
```
# Model Architecture

model.build((padded_train_sequences.shape))
model.summary()
```

```
Model: "sequential"
```

```
┏━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳
━━━━━━━━━━━━━━━━━┓
┃ Layer (type)                          ┃ Output Shape                 ┃
Param # ┃
┡━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇
━━━━━━━━━━━━━━━━━┩
│ embedding (Embedding)                 │ (12044, 73, 100)             │
3,461,000 │
├───────────────────────────────────────┼──────────────────────────────┼
```

```
───────────────────┤
│ spatial_dropout1d (SpatialDropout1D) │ (12044, 73, 100)              │
0 │
├──────────────────────────────────────┼───────────────────────────────┼
───────────────────┤
│ bidirectional (Bidirectional)        │ (12044, 73, 128)              │
84,480 │
├──────────────────────────────────────┼───────────────────────────────┼
───────────────────┤
│ batch_normalization                  │ (12044, 73, 128)              │
512 │
│ (BatchNormalization)                 │                               │
│
├──────────────────────────────────────┼───────────────────────────────┼
───────────────────┤
│ dropout (Dropout)                    │ (12044, 73, 128)              │
0 │
├──────────────────────────────────────┼───────────────────────────────┼
───────────────────┤
│ bidirectional_1 (Bidirectional)      │ (12044, 64)                   │
41,216 │
├──────────────────────────────────────┼───────────────────────────────┼
───────────────────┤
│ batch_normalization_1                │ (12044, 64)                   │
256 │
│ (BatchNormalization)                 │                               │
│
├──────────────────────────────────────┼───────────────────────────────┼
───────────────────┤
│ dropout_1 (Dropout)                  │ (12044, 64)                   │
0 │
├──────────────────────────────────────┼───────────────────────────────┼
───────────────────┤
│ dense (Dense)                        │ (12044, 24)                   │
1,560 │
├──────────────────────────────────────┼───────────────────────────────┼
───────────────────┤
│ batch_normalization_2                │ (12044, 24)                   │
96 │
│ (BatchNormalization)                 │                               │
│
├──────────────────────────────────────┼───────────────────────────────┼
───────────────────┤
│ dropout_2 (Dropout)                  │ (12044, 24)                   │
0 │
├──────────────────────────────────────┼───────────────────────────────┼
───────────────────┤
│ dense_1 (Dense)                      │ (12044, 16)                   │
400 │
```

```
├──────────────────────────────────────┼─────────────────────────────┼
─────────────────┤
│ batch_normalization_3                │ (12044, 16)                 │
64 │
│ (BatchNormalization)                 │                             │
│
├──────────────────────────────────────┼─────────────────────────────┼
─────────────────┤
│ dropout_3 (Dropout)                  │ (12044, 16)                 │
0 │
├──────────────────────────────────────┼─────────────────────────────┼
─────────────────┤
│ dense_2 (Dense)                      │ (12044, 17)                 │
289 │
└──────────────────────────────────────┴─────────────────────────────┴
─────────────────┘

 Total params: 3,589,873 (13.69 MB)

 Trainable params: 3,589,409 (13.69 MB)

 Non-trainable params: 464 (1.81 KB)

# Visualizing the model architecture

plot_model(model, show_shapes=True, show_layer_names=True, dpi=90)
```


embedding (Embedding)
Input shape: (12044, 73)
Output shape: (12044, 73, 100)
spatial_dropout1d (SpatialDropout1D)
Input shape: (12044, 73, 100)
Output shape: (12044, 73, 100)
bidirectional (Bidirectional)
Input shape: (12044, 73, 100)
Output shape: (12044, 73, 128)
batch_normalization (BatchNormalization)
Input shape: (12044, 73, 128)
Output shape: (12044, 73, 128)
dropout (Dropout)
Input shape: (12044, 73, 128)
Output shape: (12044, 73, 128)

Setting up the relevant training elements and tuning the hyperparameters

verbose=1: povides a detailed output with progress bars, metrics for each epoch, and any additional callback messages

verbose=0: no progress bars or messages will be shown

verbose=2: shows only one line per epoch with epoch and metric updates but no progress bar

```
y_train_original = np.argmax(y_train, axis=1)  # Converting one-hot
encoded y_train back to label form

# Compute class weights
class_weights = compute_class_weight('balanced',
classes=np.unique(y_train_original), y=y_train_original) # calculates
the weight for each class based on its frequency
class_weights = dict(enumerate(class_weights))

num_epochs = 200 # setting up epoch numbers

reduce_lr = ReduceLROnPlateau( # reduces the learning rate f the
val_loss does not improve
    monitor='val_loss',
    factor=0.2, # reduces the learning rate by a factor of 0.2
if......
    patience=3, # ......if the val_loss does not improve for 3
consecutive epochs
    min_lr=1e-6,# the minimum threshold for the learning rate
    verbose=1
)

checkpoint = ModelCheckpoint( # saves the model weights whenever
val_accuracy improves
    'best_model.keras',
    monitor='val_accuracy',
    save_best_only=True, # ensures that only the best weights are
saved based on validation accuracy
    mode='max',
    verbose=1
)

early_stop = EarlyStopping(monitor='val_loss', patience=10) # monitors
the val_loss and stops training if it doesn’t improve for 10
consecutive epochs

model.compile(loss = 'categorical_crossentropy', # calculates the loss
by comparing the model’s predicted probabilities to the one-hot-
encoded true labels
              optimizer=Adam(learning_rate=0.000001, clipnorm=1.0), #
```

```
clipnorm=1.0 prevents the gradients from growing too large by capping
their norm to 1
              metrics = ['accuracy']) # evaluation metric during
training

history = model.fit(padded_train_sequences, # input sequences
                    y_train, # output labels of those input sequences
                    validation_data=(padded_val_sequences, y_val), #
validation data and labels
                    epochs=num_epochs, #epoch numbers
                    class_weight=class_weights, # counteracts class
imbalance by adjusting the model’s loss calculation by giving
different weights to each class
                    callbacks =[reduce_lr, early_stop, checkpoint], #
applies the learning rate scheduler, model checkpointing, and early
stopping during training
                    batch_size=32, # the number of samples processed
before updating the model weights
                    verbose=1)
```

```
Epoch 1/200
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - accuracy: 0.0800 - loss:
5.9269
Epoch 1: val_accuracy improved from -inf to 0.06777, saving model to
best_model.keras
377/377 ━━━━━━━━━━━━━━━━━━━━ 19s 26ms/step - accuracy: 0.0800 - loss:
5.9272 - val_accuracy: 0.0678 - val_loss: 5.6195 - learning_rate:
1.0000e-06
Epoch 2/200
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - accuracy: 0.0701 - loss:
6.2302
Epoch 2: val_accuracy improved from 0.06777 to 0.09169, saving model
to best_model.keras
377/377 ━━━━━━━━━━━━━━━━━━━━ 9s 24ms/step - accuracy: 0.0701 - loss:
6.2297 - val_accuracy: 0.0917 - val_loss: 5.6237 - learning_rate:
1.0000e-06
Epoch 3/200
376/377 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - accuracy: 0.0680 - loss:
5.7989
Epoch 3: val_accuracy improved from 0.09169 to 0.09236, saving model
to best_model.keras
377/377 ━━━━━━━━━━━━━━━━━━━━ 9s 24ms/step - accuracy: 0.0681 - loss:
5.7997 - val_accuracy: 0.0924 - val_loss: 5.6115 - learning_rate:
1.0000e-06
Epoch 4/200
375/377 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - accuracy: 0.0754 - loss:
5.8490
Epoch 4: val_accuracy improved from 0.09236 to 0.09568, saving model
to best_model.keras
377/377 ━━━━━━━━━━━━━━━━━━━━ 9s 24ms/step - accuracy: 0.0754 - loss:
```

```
5.8502 - val_accuracy: 0.0957 - val_loss: 5.6134 - learning_rate:
1.0000e-06
Epoch 5/200
375/377 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - accuracy: 0.0758 - loss:
6.1871
Epoch 5: val_accuracy improved from 0.09568 to 0.09635, saving model
to best_model.keras
377/377 ━━━━━━━━━━━━━━━━━━━━ 9s 24ms/step - accuracy: 0.0758 - loss:
6.1858 - val_accuracy: 0.0963 - val_loss: 5.6048 - learning_rate:
1.0000e-06
Epoch 6/200
375/377 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - accuracy: 0.0692 - loss:
6.0434
Epoch 6: val_accuracy did not improve from 0.09635
377/377 ━━━━━━━━━━━━━━━━━━━━ 9s 23ms/step - accuracy: 0.0693 - loss:
6.0434 - val_accuracy: 0.0930 - val_loss: 5.6114 - learning_rate:
1.0000e-06
Epoch 7/200
375/377 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - accuracy: 0.0745 - loss:
5.8648
Epoch 7: val_accuracy did not improve from 0.09635
377/377 ━━━━━━━━━━━━━━━━━━━━ 9s 23ms/step - accuracy: 0.0744 - loss:
5.8663 - val_accuracy: 0.0917 - val_loss: 5.6102 - learning_rate:
1.0000e-06
Epoch 8/200
376/377 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - accuracy: 0.0693 - loss:
6.1075
Epoch 8: val_accuracy did not improve from 0.09635
377/377 ━━━━━━━━━━━━━━━━━━━━ 9s 23ms/step - accuracy: 0.0693 - loss:
6.1074 - val_accuracy: 0.0937 - val_loss: 5.6049 - learning_rate:
1.0000e-06
Epoch 9/200
375/377 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - accuracy: 0.0727 - loss:
5.9698
Epoch 9: val_accuracy did not improve from 0.09635
377/377 ━━━━━━━━━━━━━━━━━━━━ 9s 23ms/step - accuracy: 0.0727 - loss:
5.9703 - val_accuracy: 0.0963 - val_loss: 5.6050 - learning_rate:
1.0000e-06
Epoch 10/200
375/377 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - accuracy: 0.0694 - loss:
6.0968
Epoch 10: val_accuracy did not improve from 0.09635
377/377 ━━━━━━━━━━━━━━━━━━━━ 9s 23ms/step - accuracy: 0.0695 - loss:
6.0959 - val_accuracy: 0.0930 - val_loss: 5.5978 - learning_rate:
1.0000e-06
Epoch 11/200
375/377 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - accuracy: 0.0749 - loss:
6.0765
Epoch 11: val_accuracy improved from 0.09635 to 0.09701, saving model
```

```
to best_model.keras
377/377 ━━━━━━━━━━━━━━━━━━━━ 9s 24ms/step - accuracy: 0.0749 - loss:
6.0762 - val_accuracy: 0.0970 - val_loss: 5.5843 - learning_rate:
1.0000e-06
Epoch 12/200
376/377 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - accuracy: 0.0775 - loss:
6.1979
Epoch 12: val_accuracy did not improve from 0.09701
377/377 ━━━━━━━━━━━━━━━━━━━━ 9s 23ms/step - accuracy: 0.0775 - loss:
6.1969 - val_accuracy: 0.0970 - val_loss: 5.5928 - learning_rate:
1.0000e-06
Epoch 13/200
376/377 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - accuracy: 0.0694 - loss:
6.2052
Epoch 13: val_accuracy did not improve from 0.09701
377/377 ━━━━━━━━━━━━━━━━━━━━ 9s 23ms/step - accuracy: 0.0694 - loss:
6.2041 - val_accuracy: 0.0944 - val_loss: 5.5908 - learning_rate:
1.0000e-06
Epoch 14/200
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - accuracy: 0.0753 - loss:
5.7443
Epoch 14: val_accuracy improved from 0.09701 to 0.10033, saving model
to best_model.keras
377/377 ━━━━━━━━━━━━━━━━━━━━ 9s 24ms/step - accuracy: 0.0753 - loss:
5.7447 - val_accuracy: 0.1003 - val_loss: 5.5596 - learning_rate:
1.0000e-06
Epoch 15/200
376/377 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - accuracy: 0.0732 - loss:
5.9533
Epoch 15: val_accuracy did not improve from 0.10033
377/377 ━━━━━━━━━━━━━━━━━━━━ 9s 24ms/step - accuracy: 0.0732 - loss:
5.9537 - val_accuracy: 0.0963 - val_loss: 5.5844 - learning_rate:
1.0000e-06
Epoch 16/200
375/377 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - accuracy: 0.0675 - loss:
5.9373
Epoch 16: val_accuracy did not improve from 0.10033
377/377 ━━━━━━━━━━━━━━━━━━━━ 9s 23ms/step - accuracy: 0.0675 - loss:
5.9376 - val_accuracy: 0.0990 - val_loss: 5.5709 - learning_rate:
1.0000e-06
Epoch 17/200
375/377 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - accuracy: 0.0793 - loss:
5.7460
Epoch 17: val_accuracy did not improve from 0.10033
377/377 ━━━━━━━━━━━━━━━━━━━━ 9s 23ms/step - accuracy: 0.0792 - loss:
5.7481 - val_accuracy: 0.0910 - val_loss: 5.5784 - learning_rate:
1.0000e-06
Epoch 18/200
376/377 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - accuracy: 0.0699 - loss:
```

```
5.8579
Epoch 18: val_accuracy improved from 0.10033 to 0.10299, saving model
to best_model.keras
377/377 ━━━━━━━━━━━━━━━━━━━━ 9s 24ms/step - accuracy: 0.0699 - loss:
5.8582 - val_accuracy: 0.1030 - val_loss: 5.5482 - learning_rate:
1.0000e-06
Epoch 19/200
376/377 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - accuracy: 0.0690 - loss:
6.1331
Epoch 19: val_accuracy did not improve from 0.10299
377/377 ━━━━━━━━━━━━━━━━━━━━ 9s 23ms/step - accuracy: 0.0691 - loss:
6.1324 - val_accuracy: 0.0950 - val_loss: 5.5581 - learning_rate:
1.0000e-06
Epoch 20/200
376/377 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - accuracy: 0.0695 - loss:
6.1251
Epoch 20: val_accuracy did not improve from 0.10299
377/377 ━━━━━━━━━━━━━━━━━━━━ 9s 23ms/step - accuracy: 0.0695 - loss:
6.1245 - val_accuracy: 0.0970 - val_loss: 5.5540 - learning_rate:
1.0000e-06
Epoch 21/200
375/377 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - accuracy: 0.0691 - loss:
6.0629
Epoch 21: val_accuracy did not improve from 0.10299
377/377 ━━━━━━━━━━━━━━━━━━━━ 9s 23ms/step - accuracy: 0.0691 - loss:
6.0626 - val_accuracy: 0.0970 - val_loss: 5.5472 - learning_rate:
1.0000e-06
Epoch 22/200
375/377 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - accuracy: 0.0724 - loss:
6.0155
Epoch 22: val_accuracy did not improve from 0.10299
377/377 ━━━━━━━━━━━━━━━━━━━━ 9s 23ms/step - accuracy: 0.0724 - loss:
6.0148 - val_accuracy: 0.0957 - val_loss: 5.5629 - learning_rate:
1.0000e-06
Epoch 23/200
375/377 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - accuracy: 0.0717 - loss:
5.9334
Epoch 23: val_accuracy did not improve from 0.10299
377/377 ━━━━━━━━━━━━━━━━━━━━ 9s 23ms/step - accuracy: 0.0717 - loss:
5.9342 - val_accuracy: 0.0937 - val_loss: 5.5587 - learning_rate:
1.0000e-06
Epoch 24/200
375/377 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - accuracy: 0.0775 - loss:
6.0551
Epoch 24: val_accuracy did not improve from 0.10299
377/377 ━━━━━━━━━━━━━━━━━━━━ 9s 23ms/step - accuracy: 0.0774 - loss:
6.0537 - val_accuracy: 0.0990 - val_loss: 5.5427 - learning_rate:
1.0000e-06
Epoch 25/200
```

```
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - accuracy: 0.0732 - loss:
5.8312
Epoch 25: val_accuracy did not improve from 0.10299
377/377 ━━━━━━━━━━━━━━━━━━━━ 9s 23ms/step - accuracy: 0.0732 - loss:
5.8315 - val_accuracy: 0.0983 - val_loss: 5.5382 - learning_rate:
1.0000e-06
Epoch 26/200
375/377 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - accuracy: 0.0748 - loss:
5.7763
Epoch 26: val_accuracy did not improve from 0.10299
377/377 ━━━━━━━━━━━━━━━━━━━━ 9s 23ms/step - accuracy: 0.0748 - loss:
5.7779 - val_accuracy: 0.0997 - val_loss: 5.5374 - learning_rate:
1.0000e-06
Epoch 27/200
375/377 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - accuracy: 0.0683 - loss:
5.9559
Epoch 27: val_accuracy did not improve from 0.10299
377/377 ━━━━━━━━━━━━━━━━━━━━ 9s 23ms/step - accuracy: 0.0683 - loss:
5.9553 - val_accuracy: 0.0957 - val_loss: 5.5385 - learning_rate:
1.0000e-06
Epoch 28/200
376/377 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - accuracy: 0.0691 - loss:
5.8738
Epoch 28: val_accuracy did not improve from 0.10299
377/377 ━━━━━━━━━━━━━━━━━━━━ 9s 23ms/step - accuracy: 0.0691 - loss:
5.8739 - val_accuracy: 0.0983 - val_loss: 5.5349 - learning_rate:
1.0000e-06
Epoch 29/200
375/377 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - accuracy: 0.0748 - loss:
5.9970
Epoch 29: val_accuracy did not improve from 0.10299
377/377 ━━━━━━━━━━━━━━━━━━━━ 9s 24ms/step - accuracy: 0.0748 - loss:
5.9968 - val_accuracy: 0.1010 - val_loss: 5.5203 - learning_rate:
1.0000e-06
Epoch 30/200
376/377 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - accuracy: 0.0718 - loss:
5.6542
Epoch 30: val_accuracy did not improve from 0.10299
377/377 ━━━━━━━━━━━━━━━━━━━━ 9s 23ms/step - accuracy: 0.0718 - loss:
5.6552 - val_accuracy: 0.1010 - val_loss: 5.5189 - learning_rate:
1.0000e-06
Epoch 31/200
375/377 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - accuracy: 0.0665 - loss:
5.8524
Epoch 31: val_accuracy did not improve from 0.10299
377/377 ━━━━━━━━━━━━━━━━━━━━ 9s 23ms/step - accuracy: 0.0665 - loss:
5.8534 - val_accuracy: 0.0957 - val_loss: 5.5296 - learning_rate:
1.0000e-06
Epoch 32/200
```

```
375/377 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - accuracy: 0.0740 - loss:
5.6789
Epoch 32: val_accuracy did not improve from 0.10299
377/377 ━━━━━━━━━━━━━━━━━━━━ 9s 23ms/step - accuracy: 0.0740 - loss:
5.6804 - val_accuracy: 0.0957 - val_loss: 5.5225 - learning_rate:
1.0000e-06
Epoch 33/200
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - accuracy: 0.0719 - loss:
5.9983
Epoch 33: val_accuracy did not improve from 0.10299
377/377 ━━━━━━━━━━━━━━━━━━━━ 9s 24ms/step - accuracy: 0.0719 - loss:
5.9983 - val_accuracy: 0.0970 - val_loss: 5.5162 - learning_rate:
1.0000e-06
Epoch 34/200
375/377 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - accuracy: 0.0768 - loss:
5.9427
Epoch 34: val_accuracy did not improve from 0.10299
377/377 ━━━━━━━━━━━━━━━━━━━━ 9s 23ms/step - accuracy: 0.0768 - loss:
5.9424 - val_accuracy: 0.0970 - val_loss: 5.5176 - learning_rate:
1.0000e-06
Epoch 35/200
375/377 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - accuracy: 0.0681 - loss:
5.7899
Epoch 35: val_accuracy did not improve from 0.10299
377/377 ━━━━━━━━━━━━━━━━━━━━ 9s 23ms/step - accuracy: 0.0681 - loss:
5.7906 - val_accuracy: 0.0977 - val_loss: 5.5068 - learning_rate:
1.0000e-06
Epoch 36/200
375/377 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - accuracy: 0.0669 - loss:
5.9779
Epoch 36: val_accuracy improved from 0.10299 to 0.10565, saving model
to best_model.keras
377/377 ━━━━━━━━━━━━━━━━━━━━ 9s 24ms/step - accuracy: 0.0669 - loss:
5.9776 - val_accuracy: 0.1056 - val_loss: 5.4833 - learning_rate:
1.0000e-06
Epoch 37/200
375/377 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - accuracy: 0.0621 - loss:
5.8314
Epoch 37: val_accuracy did not improve from 0.10565
377/377 ━━━━━━━━━━━━━━━━━━━━ 9s 23ms/step - accuracy: 0.0622 - loss:
5.8317 - val_accuracy: 0.1043 - val_loss: 5.4921 - learning_rate:
1.0000e-06
Epoch 38/200
376/377 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - accuracy: 0.0742 - loss:
5.9117
Epoch 38: val_accuracy improved from 0.10565 to 0.10897, saving model
to best_model.keras
377/377 ━━━━━━━━━━━━━━━━━━━━ 9s 24ms/step - accuracy: 0.0742 - loss:
5.9116 - val_accuracy: 0.1090 - val_loss: 5.4891 - learning_rate:
```

```
1.0000e-06
Epoch 39/200
376/377 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - accuracy: 0.0713 - loss:
6.0930
Epoch 39: val_accuracy did not improve from 0.10897
377/377 ━━━━━━━━━━━━━━━━━━━━ 9s 23ms/step - accuracy: 0.0713 - loss:
6.0923 - val_accuracy: 0.0970 - val_loss: 5.4980 - learning_rate:
1.0000e-06
Epoch 40/200
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - accuracy: 0.0764 - loss:
5.9408
Epoch 40: val_accuracy did not improve from 0.10897
377/377 ━━━━━━━━━━━━━━━━━━━━ 9s 24ms/step - accuracy: 0.0764 - loss:
5.9405 - val_accuracy: 0.0977 - val_loss: 5.4932 - learning_rate:
1.0000e-06
Epoch 41/200
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - accuracy: 0.0699 - loss:
5.6831
Epoch 41: val_accuracy did not improve from 0.10897
377/377 ━━━━━━━━━━━━━━━━━━━━ 9s 23ms/step - accuracy: 0.0699 - loss:
5.6837 - val_accuracy: 0.1023 - val_loss: 5.4835 - learning_rate:
1.0000e-06
Epoch 42/200
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - accuracy: 0.0731 - loss:
5.8205
Epoch 42: val_accuracy did not improve from 0.10897
377/377 ━━━━━━━━━━━━━━━━━━━━ 9s 23ms/step - accuracy: 0.0731 - loss:
5.8206 - val_accuracy: 0.1023 - val_loss: 5.4824 - learning_rate:
1.0000e-06
Epoch 43/200
375/377 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - accuracy: 0.0751 - loss:
5.9834
Epoch 43: val_accuracy did not improve from 0.10897
377/377 ━━━━━━━━━━━━━━━━━━━━ 9s 23ms/step - accuracy: 0.0751 - loss:
5.9827 - val_accuracy: 0.0970 - val_loss: 5.4893 - learning_rate:
1.0000e-06
Epoch 44/200
376/377 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - accuracy: 0.0712 - loss:
6.1470
Epoch 44: val_accuracy did not improve from 0.10897
377/377 ━━━━━━━━━━━━━━━━━━━━ 9s 23ms/step - accuracy: 0.0712 - loss:
6.1450 - val_accuracy: 0.0963 - val_loss: 5.4852 - learning_rate:
1.0000e-06
Epoch 45/200
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - accuracy: 0.0725 - loss:
5.8294
Epoch 45: val_accuracy did not improve from 0.10897
377/377 ━━━━━━━━━━━━━━━━━━━━ 9s 23ms/step - accuracy: 0.0725 - loss:
5.8294 - val_accuracy: 0.0950 - val_loss: 5.4718 - learning_rate:
```

```
1.0000e-06
Epoch 46/200
376/377 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - accuracy: 0.0786 - loss:
5.8727
Epoch 46: val_accuracy did not improve from 0.10897
377/377 ━━━━━━━━━━━━━━━━━━━━ 9s 23ms/step - accuracy: 0.0786 - loss:
5.8726 - val_accuracy: 0.0963 - val_loss: 5.4729 - learning_rate:
1.0000e-06
Epoch 47/200
375/377 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - accuracy: 0.0745 - loss:
5.6847
Epoch 47: val_accuracy did not improve from 0.10897
377/377 ━━━━━━━━━━━━━━━━━━━━ 9s 24ms/step - accuracy: 0.0745 - loss:
5.6858 - val_accuracy: 0.0990 - val_loss: 5.4620 - learning_rate:
1.0000e-06
Epoch 48/200
376/377 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - accuracy: 0.0704 - loss:
5.8436
Epoch 48: val_accuracy did not improve from 0.10897
377/377 ━━━━━━━━━━━━━━━━━━━━ 9s 23ms/step - accuracy: 0.0704 - loss:
5.8437 - val_accuracy: 0.1056 - val_loss: 5.4486 - learning_rate:
1.0000e-06
Epoch 49/200
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - accuracy: 0.0696 - loss:
5.6795
Epoch 49: val_accuracy did not improve from 0.10897
377/377 ━━━━━━━━━━━━━━━━━━━━ 9s 23ms/step - accuracy: 0.0696 - loss:
5.6799 - val_accuracy: 0.1037 - val_loss: 5.4564 - learning_rate:
1.0000e-06
Epoch 50/200
375/377 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - accuracy: 0.0721 - loss:
5.7847
Epoch 50: val_accuracy did not improve from 0.10897
377/377 ━━━━━━━━━━━━━━━━━━━━ 9s 23ms/step - accuracy: 0.0721 - loss:
5.7855 - val_accuracy: 0.1037 - val_loss: 5.4511 - learning_rate:
1.0000e-06
Epoch 51/200
376/377 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - accuracy: 0.0684 - loss:
5.9645
Epoch 51: val_accuracy did not improve from 0.10897
377/377 ━━━━━━━━━━━━━━━━━━━━ 9s 24ms/step - accuracy: 0.0684 - loss:
5.9637 - val_accuracy: 0.0950 - val_loss: 5.4606 - learning_rate:
1.0000e-06
Epoch 52/200
376/377 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - accuracy: 0.0761 - loss:
6.0038
Epoch 52: val_accuracy did not improve from 0.10897
377/377 ━━━━━━━━━━━━━━━━━━━━ 9s 23ms/step - accuracy: 0.0761 - loss:
6.0032 - val_accuracy: 0.0997 - val_loss: 5.4582 - learning_rate:
```

```
1.0000e-06
Epoch 53/200
375/377 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - accuracy: 0.0709 - loss:
6.0259
Epoch 53: val_accuracy did not improve from 0.10897
377/377 ━━━━━━━━━━━━━━━━━━━━ 9s 23ms/step - accuracy: 0.0709 - loss:
6.0254 - val_accuracy: 0.1010 - val_loss: 5.4446 - learning_rate:
1.0000e-06
Epoch 54/200
375/377 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - accuracy: 0.0753 - loss:
5.6703
Epoch 54: val_accuracy did not improve from 0.10897
377/377 ━━━━━━━━━━━━━━━━━━━━ 9s 24ms/step - accuracy: 0.0752 - loss:
5.6721 - val_accuracy: 0.0997 - val_loss: 5.4430 - learning_rate:
1.0000e-06
Epoch 55/200
375/377 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - accuracy: 0.0720 - loss:
5.9343
Epoch 55: val_accuracy did not improve from 0.10897
377/377 ━━━━━━━━━━━━━━━━━━━━ 9s 23ms/step - accuracy: 0.0720 - loss:
5.9330 - val_accuracy: 0.1030 - val_loss: 5.4312 - learning_rate:
1.0000e-06
Epoch 56/200
376/377 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - accuracy: 0.0680 - loss:
5.8794
Epoch 56: val_accuracy did not improve from 0.10897
377/377 ━━━━━━━━━━━━━━━━━━━━ 9s 23ms/step - accuracy: 0.0680 - loss:
5.8791 - val_accuracy: 0.0977 - val_loss: 5.4401 - learning_rate:
1.0000e-06
Epoch 57/200
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - accuracy: 0.0681 - loss:
5.8009
Epoch 57: val_accuracy did not improve from 0.10897
377/377 ━━━━━━━━━━━━━━━━━━━━ 9s 23ms/step - accuracy: 0.0681 - loss:
5.8010 - val_accuracy: 0.1017 - val_loss: 5.4218 - learning_rate:
1.0000e-06
Epoch 58/200
375/377 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - accuracy: 0.0676 - loss:
5.9505
Epoch 58: val_accuracy did not improve from 0.10897
377/377 ━━━━━━━━━━━━━━━━━━━━ 9s 24ms/step - accuracy: 0.0676 - loss:
5.9500 - val_accuracy: 0.1063 - val_loss: 5.4194 - learning_rate:
1.0000e-06
Epoch 59/200
375/377 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - accuracy: 0.0678 - loss:
5.7865
Epoch 59: val_accuracy did not improve from 0.10897
377/377 ━━━━━━━━━━━━━━━━━━━━ 9s 23ms/step - accuracy: 0.0678 - loss:
5.7866 - val_accuracy: 0.1003 - val_loss: 5.4241 - learning_rate:
```

```
1.0000e-06
Epoch 60/200
375/377 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - accuracy: 0.0690 - loss:
5.7345
Epoch 60: val_accuracy did not improve from 0.10897
377/377 ━━━━━━━━━━━━━━━━━━━━ 9s 23ms/step - accuracy: 0.0690 - loss:
5.7352 - val_accuracy: 0.1030 - val_loss: 5.4196 - learning_rate:
1.0000e-06
Epoch 61/200
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - accuracy: 0.0729 - loss:
6.0216
Epoch 61: val_accuracy did not improve from 0.10897
377/377 ━━━━━━━━━━━━━━━━━━━━ 9s 23ms/step - accuracy: 0.0729 - loss:
6.0212 - val_accuracy: 0.0997 - val_loss: 5.4209 - learning_rate:
1.0000e-06
Epoch 62/200
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - accuracy: 0.0727 - loss:
5.8970
Epoch 62: val_accuracy did not improve from 0.10897
377/377 ━━━━━━━━━━━━━━━━━━━━ 9s 24ms/step - accuracy: 0.0727 - loss:
5.8969 - val_accuracy: 0.0977 - val_loss: 5.4148 - learning_rate:
1.0000e-06
Epoch 63/200
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - accuracy: 0.0680 - loss:
6.0560
Epoch 63: val_accuracy did not improve from 0.10897
377/377 ━━━━━━━━━━━━━━━━━━━━ 9s 23ms/step - accuracy: 0.0680 - loss:
6.0552 - val_accuracy: 0.1037 - val_loss: 5.4166 - learning_rate:
1.0000e-06
Epoch 64/200
376/377 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - accuracy: 0.0686 - loss:
6.3347
Epoch 64: val_accuracy did not improve from 0.10897
377/377 ━━━━━━━━━━━━━━━━━━━━ 9s 23ms/step - accuracy: 0.0686 - loss:
6.3322 - val_accuracy: 0.1037 - val_loss: 5.4068 - learning_rate:
1.0000e-06
Epoch 65/200
376/377 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - accuracy: 0.0762 - loss:
5.5755
Epoch 65: val_accuracy did not improve from 0.10897
377/377 ━━━━━━━━━━━━━━━━━━━━ 9s 24ms/step - accuracy: 0.0762 - loss:
5.5761 - val_accuracy: 0.1030 - val_loss: 5.4048 - learning_rate:
1.0000e-06
Epoch 66/200
375/377 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - accuracy: 0.0779 - loss:
5.7830
Epoch 66: val_accuracy did not improve from 0.10897
377/377 ━━━━━━━━━━━━━━━━━━━━ 9s 23ms/step - accuracy: 0.0778 - loss:
5.7832 - val_accuracy: 0.1056 - val_loss: 5.3953 - learning_rate:
```

```
1.0000e-06
Epoch 67/200
375/377 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - accuracy: 0.0716 - loss:
5.8933
Epoch 67: val_accuracy did not improve from 0.10897
377/377 ━━━━━━━━━━━━━━━━━━━━ 9s 23ms/step - accuracy: 0.0716 - loss:
5.8933 - val_accuracy: 0.1037 - val_loss: 5.4068 - learning_rate:
1.0000e-06
Epoch 68/200
375/377 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - accuracy: 0.0722 - loss:
6.0004
Epoch 68: val_accuracy did not improve from 0.10897
377/377 ━━━━━━━━━━━━━━━━━━━━ 9s 23ms/step - accuracy: 0.0722 - loss:
5.9986 - val_accuracy: 0.1070 - val_loss: 5.3952 - learning_rate:
1.0000e-06
Epoch 69/200
375/377 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - accuracy: 0.0749 - loss:
5.8561
Epoch 69: val_accuracy did not improve from 0.10897
377/377 ━━━━━━━━━━━━━━━━━━━━ 9s 24ms/step - accuracy: 0.0749 - loss:
5.8561 - val_accuracy: 0.1056 - val_loss: 5.3936 - learning_rate:
1.0000e-06
Epoch 70/200
375/377 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - accuracy: 0.0767 - loss:
6.1035
Epoch 70: val_accuracy did not improve from 0.10897
377/377 ━━━━━━━━━━━━━━━━━━━━ 9s 23ms/step - accuracy: 0.0767 - loss:
6.1010 - val_accuracy: 0.1056 - val_loss: 5.3848 - learning_rate:
1.0000e-06
Epoch 71/200
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - accuracy: 0.0719 - loss:
5.8210
Epoch 71: val_accuracy did not improve from 0.10897
377/377 ━━━━━━━━━━━━━━━━━━━━ 9s 23ms/step - accuracy: 0.0719 - loss:
5.8210 - val_accuracy: 0.1037 - val_loss: 5.3901 - learning_rate:
1.0000e-06
Epoch 72/200
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - accuracy: 0.0674 - loss:
5.5066
Epoch 72: val_accuracy did not improve from 0.10897
377/377 ━━━━━━━━━━━━━━━━━━━━ 9s 24ms/step - accuracy: 0.0674 - loss:
5.5074 - val_accuracy: 0.1090 - val_loss: 5.3821 - learning_rate:
1.0000e-06
Epoch 73/200
375/377 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - accuracy: 0.0695 - loss:
5.7452
Epoch 73: val_accuracy did not improve from 0.10897
377/377 ━━━━━━━━━━━━━━━━━━━━ 9s 23ms/step - accuracy: 0.0695 - loss:
5.7456 - val_accuracy: 0.1076 - val_loss: 5.3722 - learning_rate:
1.0000e-06
```

```
Epoch 74/200
375/377 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - accuracy: 0.0729 - loss:
5.5909
Epoch 74: val_accuracy improved from 0.10897 to 0.11096, saving model
to best_model.keras
377/377 ━━━━━━━━━━━━━━━━━━━━ 9s 24ms/step - accuracy: 0.0729 - loss:
5.5925 - val_accuracy: 0.1110 - val_loss: 5.3677 - learning_rate:
1.0000e-06
Epoch 75/200
375/377 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - accuracy: 0.0728 - loss:
5.7315
Epoch 75: val_accuracy improved from 0.11096 to 0.11163, saving model
to best_model.keras
377/377 ━━━━━━━━━━━━━━━━━━━━ 9s 24ms/step - accuracy: 0.0727 - loss:
5.7319 - val_accuracy: 0.1116 - val_loss: 5.3700 - learning_rate:
1.0000e-06
Epoch 76/200
375/377 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - accuracy: 0.0709 - loss:
5.9147
Epoch 76: val_accuracy improved from 0.11163 to 0.11296, saving model
to best_model.keras
377/377 ━━━━━━━━━━━━━━━━━━━━ 9s 24ms/step - accuracy: 0.0709 - loss:
5.9137 - val_accuracy: 0.1130 - val_loss: 5.3585 - learning_rate:
1.0000e-06
Epoch 77/200
375/377 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - accuracy: 0.0716 - loss:
5.5600
Epoch 77: val_accuracy did not improve from 0.11296
377/377 ━━━━━━━━━━━━━━━━━━━━ 9s 23ms/step - accuracy: 0.0716 - loss:
5.5617 - val_accuracy: 0.1070 - val_loss: 5.3645 - learning_rate:
1.0000e-06
Epoch 78/200
375/377 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - accuracy: 0.0763 - loss:
5.9382
Epoch 78: val_accuracy did not improve from 0.11296
377/377 ━━━━━━━━━━━━━━━━━━━━ 9s 23ms/step - accuracy: 0.0763 - loss:
5.9365 - val_accuracy: 0.1130 - val_loss: 5.3572 - learning_rate:
1.0000e-06
Epoch 79/200
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - accuracy: 0.0693 - loss:
5.9053
Epoch 79: val_accuracy did not improve from 0.11296
377/377 ━━━━━━━━━━━━━━━━━━━━ 9s 24ms/step - accuracy: 0.0693 - loss:
5.9050 - val_accuracy: 0.1063 - val_loss: 5.3606 - learning_rate:
1.0000e-06
Epoch 80/200
375/377 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - accuracy: 0.0678 - loss:
5.6838
Epoch 80: val_accuracy did not improve from 0.11296
```

```
377/377 ━━━━━━━━━━━━━━━━━━━━ 9s 23ms/step - accuracy: 0.0679 - loss:
5.6840 - val_accuracy: 0.1083 - val_loss: 5.3508 - learning_rate:
1.0000e-06
Epoch 81/200
375/377 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - accuracy: 0.0710 - loss:
5.8004
Epoch 81: val_accuracy did not improve from 0.11296
377/377 ━━━━━━━━━━━━━━━━━━━━ 9s 23ms/step - accuracy: 0.0710 - loss:
5.8001 - val_accuracy: 0.1090 - val_loss: 5.3487 - learning_rate:
1.0000e-06
Epoch 82/200
376/377 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - accuracy: 0.0710 - loss:
5.6632
Epoch 82: val_accuracy did not improve from 0.11296
377/377 ━━━━━━━━━━━━━━━━━━━━ 9s 23ms/step - accuracy: 0.0710 - loss:
5.6632 - val_accuracy: 0.1076 - val_loss: 5.3464 - learning_rate:
1.0000e-06
Epoch 83/200
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - accuracy: 0.0780 - loss:
5.6700
Epoch 83: val_accuracy did not improve from 0.11296
377/377 ━━━━━━━━━━━━━━━━━━━━ 9s 24ms/step - accuracy: 0.0780 - loss:
5.6702 - val_accuracy: 0.1056 - val_loss: 5.3565 - learning_rate:
1.0000e-06
Epoch 84/200
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - accuracy: 0.0810 - loss:
5.4278
Epoch 84: val_accuracy did not improve from 0.11296
377/377 ━━━━━━━━━━━━━━━━━━━━ 9s 23ms/step - accuracy: 0.0810 - loss:
5.4285 - val_accuracy: 0.1076 - val_loss: 5.3501 - learning_rate:
1.0000e-06
Epoch 85/200
375/377 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - accuracy: 0.0733 - loss:
5.9030
Epoch 85: val_accuracy did not improve from 0.11296
377/377 ━━━━━━━━━━━━━━━━━━━━ 9s 23ms/step - accuracy: 0.0732 - loss:
5.9016 - val_accuracy: 0.1116 - val_loss: 5.3322 - learning_rate:
1.0000e-06
Epoch 86/200
376/377 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - accuracy: 0.0708 - loss:
5.5902
Epoch 86: val_accuracy did not improve from 0.11296
377/377 ━━━━━━━━━━━━━━━━━━━━ 9s 23ms/step - accuracy: 0.0708 - loss:
5.5905 - val_accuracy: 0.1103 - val_loss: 5.3342 - learning_rate:
1.0000e-06
Epoch 87/200
376/377 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - accuracy: 0.0767 - loss:
5.5798
Epoch 87: val_accuracy did not improve from 0.11296
```

```
377/377 ━━━━━━━━━━━━━━━━━━━━ 9s 24ms/step - accuracy: 0.0767 - loss:
5.5805 - val_accuracy: 0.1083 - val_loss: 5.3262 - learning_rate:
1.0000e-06
Epoch 88/200
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - accuracy: 0.0738 - loss:
5.5699
Epoch 88: val_accuracy did not improve from 0.11296
377/377 ━━━━━━━━━━━━━━━━━━━━ 9s 23ms/step - accuracy: 0.0738 - loss:
5.5704 - val_accuracy: 0.1056 - val_loss: 5.3333 - learning_rate:
1.0000e-06
Epoch 89/200
375/377 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - accuracy: 0.0684 - loss:
5.5271
Epoch 89: val_accuracy did not improve from 0.11296
377/377 ━━━━━━━━━━━━━━━━━━━━ 9s 23ms/step - accuracy: 0.0684 - loss:
5.5287 - val_accuracy: 0.1123 - val_loss: 5.3254 - learning_rate:
1.0000e-06
Epoch 90/200
376/377 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - accuracy: 0.0679 - loss:
5.6026
Epoch 90: val_accuracy did not improve from 0.11296
377/377 ━━━━━━━━━━━━━━━━━━━━ 9s 24ms/step - accuracy: 0.0679 - loss:
5.6035 - val_accuracy: 0.1090 - val_loss: 5.3203 - learning_rate:
1.0000e-06
Epoch 91/200
376/377 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - accuracy: 0.0713 - loss:
5.7163
Epoch 91: val_accuracy did not improve from 0.11296
377/377 ━━━━━━━━━━━━━━━━━━━━ 9s 23ms/step - accuracy: 0.0713 - loss:
5.7162 - val_accuracy: 0.1090 - val_loss: 5.3183 - learning_rate:
1.0000e-06
Epoch 92/200
375/377 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - accuracy: 0.0678 - loss:
5.5471
Epoch 92: val_accuracy did not improve from 0.11296
377/377 ━━━━━━━━━━━━━━━━━━━━ 9s 23ms/step - accuracy: 0.0678 - loss:
5.5474 - val_accuracy: 0.1070 - val_loss: 5.3165 - learning_rate:
1.0000e-06
Epoch 93/200
375/377 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - accuracy: 0.0679 - loss:
5.7512
Epoch 93: val_accuracy did not improve from 0.11296
377/377 ━━━━━━━━━━━━━━━━━━━━ 9s 23ms/step - accuracy: 0.0679 - loss:
5.7508 - val_accuracy: 0.1070 - val_loss: 5.3173 - learning_rate:
1.0000e-06
Epoch 94/200
376/377 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - accuracy: 0.0673 - loss:
5.8086
Epoch 94: val_accuracy did not improve from 0.11296
```

```
377/377 ━━━━━━━━━━━━━━━━━━━━ 9s 24ms/step - accuracy: 0.0673 - loss:
5.8082 - val_accuracy: 0.1076 - val_loss: 5.3145 - learning_rate:
1.0000e-06
Epoch 95/200
375/377 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - accuracy: 0.0706 - loss:
5.8690
Epoch 95: val_accuracy did not improve from 0.11296
377/377 ━━━━━━━━━━━━━━━━━━━━ 9s 23ms/step - accuracy: 0.0706 - loss:
5.8671 - val_accuracy: 0.1070 - val_loss: 5.3084 - learning_rate:
1.0000e-06
Epoch 96/200
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - accuracy: 0.0718 - loss:
5.7532
Epoch 96: val_accuracy did not improve from 0.11296
377/377 ━━━━━━━━━━━━━━━━━━━━ 9s 23ms/step - accuracy: 0.0718 - loss:
5.7531 - val_accuracy: 0.1043 - val_loss: 5.3023 - learning_rate:
1.0000e-06
Epoch 97/200
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - accuracy: 0.0702 - loss:
5.6590
Epoch 97: val_accuracy did not improve from 0.11296
377/377 ━━━━━━━━━━━━━━━━━━━━ 9s 24ms/step - accuracy: 0.0702 - loss:
5.6592 - val_accuracy: 0.1096 - val_loss: 5.2952 - learning_rate:
1.0000e-06
Epoch 98/200
375/377 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - accuracy: 0.0690 - loss:
5.4302
Epoch 98: val_accuracy did not improve from 0.11296
377/377 ━━━━━━━━━━━━━━━━━━━━ 9s 23ms/step - accuracy: 0.0690 - loss:
5.4320 - val_accuracy: 0.1130 - val_loss: 5.2854 - learning_rate:
1.0000e-06
Epoch 99/200
376/377 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - accuracy: 0.0759 - loss:
5.4951
Epoch 99: val_accuracy did not improve from 0.11296
377/377 ━━━━━━━━━━━━━━━━━━━━ 9s 23ms/step - accuracy: 0.0759 - loss:
5.4962 - val_accuracy: 0.1063 - val_loss: 5.2884 - learning_rate:
1.0000e-06
Epoch 100/200
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - accuracy: 0.0726 - loss:
5.4914
Epoch 100: val_accuracy did not improve from 0.11296
377/377 ━━━━━━━━━━━━━━━━━━━━ 9s 23ms/step - accuracy: 0.0726 - loss:
5.4917 - val_accuracy: 0.0997 - val_loss: 5.3005 - learning_rate:
1.0000e-06
Epoch 101/200
376/377 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - accuracy: 0.0689 - loss:
5.5498
Epoch 101: val_accuracy did not improve from 0.11296
```

```
377/377 ━━━━━━━━━━━━━━━━━━━━ 9s 24ms/step - accuracy: 0.0689 - loss:
5.5503 - val_accuracy: 0.1056 - val_loss: 5.2901 - learning_rate:
1.0000e-06
Epoch 102/200
375/377 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - accuracy: 0.0674 - loss:
5.5509
Epoch 102: val_accuracy did not improve from 0.11296
377/377 ━━━━━━━━━━━━━━━━━━━━ 9s 23ms/step - accuracy: 0.0674 - loss:
5.5517 - val_accuracy: 0.1096 - val_loss: 5.2696 - learning_rate:
1.0000e-06
Epoch 103/200
375/377 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - accuracy: 0.0817 - loss:
5.6988
Epoch 103: val_accuracy did not improve from 0.11296
377/377 ━━━━━━━━━━━━━━━━━━━━ 9s 23ms/step - accuracy: 0.0816 - loss:
5.6988 - val_accuracy: 0.1010 - val_loss: 5.2851 - learning_rate:
1.0000e-06
Epoch 104/200
375/377 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - accuracy: 0.0702 - loss:
5.5190
Epoch 104: val_accuracy did not improve from 0.11296
377/377 ━━━━━━━━━━━━━━━━━━━━ 9s 23ms/step - accuracy: 0.0702 - loss:
5.5200 - val_accuracy: 0.1043 - val_loss: 5.2739 - learning_rate:
1.0000e-06
Epoch 105/200
376/377 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - accuracy: 0.0712 - loss:
5.5671
Epoch 105: val_accuracy did not improve from 0.11296
377/377 ━━━━━━━━━━━━━━━━━━━━ 9s 24ms/step - accuracy: 0.0712 - loss:
5.5675 - val_accuracy: 0.1063 - val_loss: 5.2767 - learning_rate:
1.0000e-06
Epoch 106/200
375/377 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - accuracy: 0.0680 - loss:
5.5977
Epoch 106: val_accuracy did not improve from 0.11296
377/377 ━━━━━━━━━━━━━━━━━━━━ 9s 23ms/step - accuracy: 0.0680 - loss:
5.5986 - val_accuracy: 0.1056 - val_loss: 5.2698 - learning_rate:
1.0000e-06
Epoch 107/200
375/377 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - accuracy: 0.0667 - loss:
5.6303
Epoch 107: val_accuracy did not improve from 0.11296
377/377 ━━━━━━━━━━━━━━━━━━━━ 9s 23ms/step - accuracy: 0.0667 - loss:
5.6299 - val_accuracy: 0.1043 - val_loss: 5.2727 - learning_rate:
1.0000e-06
Epoch 108/200
376/377 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - accuracy: 0.0720 - loss:
5.5245
Epoch 108: val_accuracy did not improve from 0.11296
```

```
377/377 ━━━━━━━━━━━━━━━━━━━━ 9s 24ms/step - accuracy: 0.0720 - loss:
5.5252 - val_accuracy: 0.1076 - val_loss: 5.2616 - learning_rate:
1.0000e-06
Epoch 109/200
375/377 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - accuracy: 0.0724 - loss:
5.6788
Epoch 109: val_accuracy did not improve from 0.11296
377/377 ━━━━━━━━━━━━━━━━━━━━ 9s 23ms/step - accuracy: 0.0724 - loss:
5.6785 - val_accuracy: 0.1043 - val_loss: 5.2697 - learning_rate:
1.0000e-06
Epoch 110/200
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - accuracy: 0.0741 - loss:
5.6181
Epoch 110: val_accuracy did not improve from 0.11296
377/377 ━━━━━━━━━━━━━━━━━━━━ 9s 23ms/step - accuracy: 0.0741 - loss:
5.6181 - val_accuracy: 0.1056 - val_loss: 5.2568 - learning_rate:
1.0000e-06
Epoch 111/200
376/377 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - accuracy: 0.0737 - loss:
5.6193
Epoch 111: val_accuracy did not improve from 0.11296
377/377 ━━━━━━━━━━━━━━━━━━━━ 9s 23ms/step - accuracy: 0.0737 - loss:
5.6196 - val_accuracy: 0.1076 - val_loss: 5.2517 - learning_rate:
1.0000e-06
Epoch 112/200
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - accuracy: 0.0706 - loss:
5.3553
Epoch 112: val_accuracy did not improve from 0.11296
377/377 ━━━━━━━━━━━━━━━━━━━━ 9s 24ms/step - accuracy: 0.0706 - loss:
5.3560 - val_accuracy: 0.1043 - val_loss: 5.2537 - learning_rate:
1.0000e-06
Epoch 113/200
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - accuracy: 0.0704 - loss:
5.5538
Epoch 113: val_accuracy did not improve from 0.11296
377/377 ━━━━━━━━━━━━━━━━━━━━ 9s 23ms/step - accuracy: 0.0704 - loss:
5.5539 - val_accuracy: 0.1030 - val_loss: 5.2565 - learning_rate:
1.0000e-06
Epoch 114/200
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - accuracy: 0.0705 - loss:
5.6452
Epoch 114: val_accuracy did not improve from 0.11296
377/377 ━━━━━━━━━━━━━━━━━━━━ 9s 23ms/step - accuracy: 0.0705 - loss:
5.6452 - val_accuracy: 0.1056 - val_loss: 5.2472 - learning_rate:
1.0000e-06
Epoch 115/200
376/377 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - accuracy: 0.0718 - loss:
5.5683
Epoch 115: val_accuracy did not improve from 0.11296
```

```
377/377 ━━━━━━━━━━━━━━━━━━━━ 9s 24ms/step - accuracy: 0.0718 - loss:
5.5689 - val_accuracy: 0.1070 - val_loss: 5.2424 - learning_rate:
1.0000e-06
Epoch 116/200
376/377 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - accuracy: 0.0704 - loss:
5.2395
Epoch 116: val_accuracy did not improve from 0.11296
377/377 ━━━━━━━━━━━━━━━━━━━━ 9s 23ms/step - accuracy: 0.0704 - loss:
5.2412 - val_accuracy: 0.1030 - val_loss: 5.2506 - learning_rate:
1.0000e-06
Epoch 117/200
376/377 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - accuracy: 0.0691 - loss:
5.6376
Epoch 117: val_accuracy did not improve from 0.11296
377/377 ━━━━━━━━━━━━━━━━━━━━ 9s 23ms/step - accuracy: 0.0691 - loss:
5.6376 - val_accuracy: 0.1076 - val_loss: 5.2303 - learning_rate:
1.0000e-06
Epoch 118/200
376/377 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - accuracy: 0.0682 - loss:
5.4958
Epoch 118: val_accuracy did not improve from 0.11296
377/377 ━━━━━━━━━━━━━━━━━━━━ 9s 23ms/step - accuracy: 0.0682 - loss:
5.4964 - val_accuracy: 0.1030 - val_loss: 5.2333 - learning_rate:
1.0000e-06
Epoch 119/200
375/377 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - accuracy: 0.0698 - loss:
5.4706
Epoch 119: val_accuracy did not improve from 0.11296
377/377 ━━━━━━━━━━━━━━━━━━━━ 9s 24ms/step - accuracy: 0.0698 - loss:
5.4718 - val_accuracy: 0.1056 - val_loss: 5.2308 - learning_rate:
1.0000e-06
Epoch 120/200
375/377 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - accuracy: 0.0710 - loss:
5.7205
Epoch 120: val_accuracy did not improve from 0.11296
377/377 ━━━━━━━━━━━━━━━━━━━━ 9s 23ms/step - accuracy: 0.0710 - loss:
5.7193 - val_accuracy: 0.1096 - val_loss: 5.2195 - learning_rate:
1.0000e-06
Epoch 121/200
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - accuracy: 0.0720 - loss:
5.7324
Epoch 121: val_accuracy did not improve from 0.11296
377/377 ━━━━━━━━━━━━━━━━━━━━ 9s 23ms/step - accuracy: 0.0720 - loss:
5.7318 - val_accuracy: 0.1090 - val_loss: 5.2256 - learning_rate:
1.0000e-06
Epoch 122/200
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - accuracy: 0.0752 - loss:
5.7803
Epoch 122: val_accuracy did not improve from 0.11296
```

```
377/377 ━━━━━━━━━━━━━━━━━━━━ 9s 23ms/step - accuracy: 0.0752 - loss:
5.7801 - val_accuracy: 0.1063 - val_loss: 5.2176 - learning_rate:
1.0000e-06
Epoch 123/200
376/377 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - accuracy: 0.0664 - loss:
5.8498
Epoch 123: val_accuracy did not improve from 0.11296
377/377 ━━━━━━━━━━━━━━━━━━━━ 9s 24ms/step - accuracy: 0.0664 - loss:
5.8484 - val_accuracy: 0.1070 - val_loss: 5.2194 - learning_rate:
1.0000e-06
Epoch 124/200
375/377 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - accuracy: 0.0737 - loss:
5.5480
Epoch 124: val_accuracy did not improve from 0.11296
377/377 ━━━━━━━━━━━━━━━━━━━━ 9s 23ms/step - accuracy: 0.0737 - loss:
5.5481 - val_accuracy: 0.1063 - val_loss: 5.2088 - learning_rate:
1.0000e-06
Epoch 125/200
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - accuracy: 0.0687 - loss:
5.5704
Epoch 125: val_accuracy did not improve from 0.11296
377/377 ━━━━━━━━━━━━━━━━━━━━ 9s 24ms/step - accuracy: 0.0687 - loss:
5.5703 - val_accuracy: 0.1083 - val_loss: 5.2020 - learning_rate:
1.0000e-06
Epoch 126/200
375/377 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - accuracy: 0.0670 - loss:
5.7465
Epoch 126: val_accuracy did not improve from 0.11296
377/377 ━━━━━━━━━━━━━━━━━━━━ 9s 24ms/step - accuracy: 0.0670 - loss:
5.7457 - val_accuracy: 0.1070 - val_loss: 5.2003 - learning_rate:
1.0000e-06
Epoch 127/200
375/377 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - accuracy: 0.0723 - loss:
5.4968
Epoch 127: val_accuracy did not improve from 0.11296
377/377 ━━━━━━━━━━━━━━━━━━━━ 9s 23ms/step - accuracy: 0.0723 - loss:
5.4975 - val_accuracy: 0.1050 - val_loss: 5.2104 - learning_rate:
1.0000e-06
Epoch 128/200
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - accuracy: 0.0758 - loss:
5.4908
Epoch 128: val_accuracy did not improve from 0.11296
377/377 ━━━━━━━━━━━━━━━━━━━━ 9s 23ms/step - accuracy: 0.0758 - loss:
5.4910 - val_accuracy: 0.1063 - val_loss: 5.2110 - learning_rate:
1.0000e-06
Epoch 129/200
375/377 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - accuracy: 0.0728 - loss:
5.2929
Epoch 129: val_accuracy did not improve from 0.11296
```

```
377/377 ━━━━━━━━━━━━━━━━━━━━ 9s 23ms/step - accuracy: 0.0728 - loss: 5.2950 - val_accuracy: 0.1043 - val_loss: 5.2055 - learning_rate: 1.0000e-06
Epoch 130/200
375/377 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - accuracy: 0.0721 - loss: 5.6797
Epoch 130: val_accuracy did not improve from 0.11296
377/377 ━━━━━━━━━━━━━━━━━━━━ 9s 24ms/step - accuracy: 0.0721 - loss: 5.6790 - val_accuracy: 0.1043 - val_loss: 5.2034 - learning_rate: 1.0000e-06
Epoch 131/200
375/377 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - accuracy: 0.0665 - loss: 5.5643
Epoch 131: val_accuracy did not improve from 0.11296
377/377 ━━━━━━━━━━━━━━━━━━━━ 9s 23ms/step - accuracy: 0.0665 - loss: 5.5645 - val_accuracy: 0.1070 - val_loss: 5.2056 - learning_rate: 1.0000e-06
Epoch 132/200
375/377 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - accuracy: 0.0685 - loss: 5.3788
Epoch 132: val_accuracy did not improve from 0.11296
377/377 ━━━━━━━━━━━━━━━━━━━━ 9s 23ms/step - accuracy: 0.0685 - loss: 5.3798 - val_accuracy: 0.1076 - val_loss: 5.1793 - learning_rate: 1.0000e-06
Epoch 133/200
375/377 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - accuracy: 0.0706 - loss: 5.3120
Epoch 133: val_accuracy did not improve from 0.11296
377/377 ━━━━━━━━━━━━━━━━━━━━ 9s 24ms/step - accuracy: 0.0706 - loss: 5.3142 - val_accuracy: 0.1076 - val_loss: 5.1806 - learning_rate: 1.0000e-06
Epoch 134/200
376/377 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - accuracy: 0.0723 - loss: 5.7261
Epoch 134: val_accuracy did not improve from 0.11296
377/377 ━━━━━━━━━━━━━━━━━━━━ 9s 23ms/step - accuracy: 0.0723 - loss: 5.7248 - val_accuracy: 0.1056 - val_loss: 5.1882 - learning_rate: 1.0000e-06
Epoch 135/200
376/377 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - accuracy: 0.0704 - loss: 5.6879
Epoch 135: val_accuracy did not improve from 0.11296
377/377 ━━━━━━━━━━━━━━━━━━━━ 9s 23ms/step - accuracy: 0.0704 - loss: 5.6874 - val_accuracy: 0.1070 - val_loss: 5.1770 - learning_rate: 1.0000e-06
Epoch 136/200
375/377 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - accuracy: 0.0754 - loss: 5.5828
Epoch 136: val_accuracy did not improve from 0.11296
```

```
377/377 ━━━━━━━━━━━━━━━━━━━━ 9s 23ms/step - accuracy: 0.0754 - loss:
5.5825 - val_accuracy: 0.1076 - val_loss: 5.1842 - learning_rate:
1.0000e-06
Epoch 137/200
375/377 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - accuracy: 0.0663 - loss:
5.4574
Epoch 137: val_accuracy did not improve from 0.11296
377/377 ━━━━━━━━━━━━━━━━━━━━ 9s 24ms/step - accuracy: 0.0664 - loss:
5.4580 - val_accuracy: 0.1050 - val_loss: 5.1790 - learning_rate:
1.0000e-06
Epoch 138/200
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - accuracy: 0.0705 - loss:
5.5481
Epoch 138: val_accuracy did not improve from 0.11296
377/377 ━━━━━━━━━━━━━━━━━━━━ 9s 23ms/step - accuracy: 0.0705 - loss:
5.5481 - val_accuracy: 0.1076 - val_loss: 5.1794 - learning_rate:
1.0000e-06
Epoch 139/200
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - accuracy: 0.0709 - loss:
5.4849
Epoch 139: val_accuracy did not improve from 0.11296
377/377 ━━━━━━━━━━━━━━━━━━━━ 9s 23ms/step - accuracy: 0.0709 - loss:
5.4849 - val_accuracy: 0.1070 - val_loss: 5.1730 - learning_rate:
1.0000e-06
Epoch 140/200
375/377 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - accuracy: 0.0691 - loss:
5.5256
Epoch 140: val_accuracy did not improve from 0.11296
377/377 ━━━━━━━━━━━━━━━━━━━━ 9s 24ms/step - accuracy: 0.0691 - loss:
5.5256 - val_accuracy: 0.1030 - val_loss: 5.1772 - learning_rate:
1.0000e-06
Epoch 141/200
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - accuracy: 0.0724 - loss:
5.4212
Epoch 141: val_accuracy did not improve from 0.11296
377/377 ━━━━━━━━━━━━━━━━━━━━ 9s 23ms/step - accuracy: 0.0724 - loss:
5.4216 - val_accuracy: 0.1083 - val_loss: 5.1718 - learning_rate:
1.0000e-06
Epoch 142/200
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - accuracy: 0.0687 - loss:
5.6404
Epoch 142: val_accuracy did not improve from 0.11296
377/377 ━━━━━━━━━━━━━━━━━━━━ 9s 23ms/step - accuracy: 0.0687 - loss:
5.6400 - val_accuracy: 0.1043 - val_loss: 5.1709 - learning_rate:
1.0000e-06
Epoch 143/200
376/377 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - accuracy: 0.0684 - loss:
5.3868
Epoch 143: val_accuracy did not improve from 0.11296
```

```
377/377 ━━━━━━━━━━━━━━━━━━━━ 9s 23ms/step - accuracy: 0.0684 - loss:
5.3874 - val_accuracy: 0.1043 - val_loss: 5.1682 - learning_rate:
1.0000e-06
Epoch 144/200
376/377 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - accuracy: 0.0748 - loss:
5.5840
Epoch 144: val_accuracy did not improve from 0.11296
377/377 ━━━━━━━━━━━━━━━━━━━━ 9s 24ms/step - accuracy: 0.0748 - loss:
5.5835 - val_accuracy: 0.1043 - val_loss: 5.1549 - learning_rate:
1.0000e-06
Epoch 145/200
375/377 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - accuracy: 0.0682 - loss:
5.3231
Epoch 145: val_accuracy did not improve from 0.11296
377/377 ━━━━━━━━━━━━━━━━━━━━ 9s 23ms/step - accuracy: 0.0682 - loss:
5.3242 - val_accuracy: 0.1070 - val_loss: 5.1520 - learning_rate:
1.0000e-06
Epoch 146/200
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - accuracy: 0.0698 - loss:
5.5400
Epoch 146: val_accuracy did not improve from 0.11296
377/377 ━━━━━━━━━━━━━━━━━━━━ 9s 23ms/step - accuracy: 0.0698 - loss:
5.5400 - val_accuracy: 0.1050 - val_loss: 5.1459 - learning_rate:
1.0000e-06
Epoch 147/200
375/377 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - accuracy: 0.0649 - loss:
5.5810
Epoch 147: val_accuracy did not improve from 0.11296
377/377 ━━━━━━━━━━━━━━━━━━━━ 9s 23ms/step - accuracy: 0.0650 - loss:
5.5802 - val_accuracy: 0.1056 - val_loss: 5.1451 - learning_rate:
1.0000e-06
Epoch 148/200
375/377 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - accuracy: 0.0715 - loss:
5.3035
Epoch 148: val_accuracy did not improve from 0.11296
377/377 ━━━━━━━━━━━━━━━━━━━━ 9s 24ms/step - accuracy: 0.0715 - loss:
5.3049 - val_accuracy: 0.1050 - val_loss: 5.1614 - learning_rate:
1.0000e-06
Epoch 149/200
375/377 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - accuracy: 0.0715 - loss:
5.5212
Epoch 149: val_accuracy did not improve from 0.11296
377/377 ━━━━━━━━━━━━━━━━━━━━ 9s 23ms/step - accuracy: 0.0715 - loss:
5.5213 - val_accuracy: 0.1037 - val_loss: 5.1449 - learning_rate:
1.0000e-06
Epoch 150/200
376/377 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - accuracy: 0.0703 - loss:
5.8396
Epoch 150: val_accuracy did not improve from 0.11296
377/377 ━━━━━━━━━━━━━━━━━━━━ 9s 23ms/step - accuracy: 0.0703 - loss:
```

```
5.8379 - val_accuracy: 0.1070 - val_loss: 5.1364 - learning_rate:
1.0000e-06
Epoch 151/200
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - accuracy: 0.0728 - loss:
5.4743
Epoch 151: val_accuracy did not improve from 0.11296
377/377 ━━━━━━━━━━━━━━━━━━━━ 9s 24ms/step - accuracy: 0.0729 - loss:
5.4744 - val_accuracy: 0.1056 - val_loss: 5.1390 - learning_rate:
1.0000e-06
Epoch 152/200
376/377 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - accuracy: 0.0736 - loss:
5.6296
Epoch 152: val_accuracy did not improve from 0.11296
377/377 ━━━━━━━━━━━━━━━━━━━━ 9s 23ms/step - accuracy: 0.0736 - loss:
5.6292 - val_accuracy: 0.1056 - val_loss: 5.1266 - learning_rate:
1.0000e-06
Epoch 153/200
375/377 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - accuracy: 0.0763 - loss:
5.4305
Epoch 153: val_accuracy did not improve from 0.11296
377/377 ━━━━━━━━━━━━━━━━━━━━ 9s 23ms/step - accuracy: 0.0763 - loss:
5.4307 - val_accuracy: 0.1070 - val_loss: 5.1194 - learning_rate:
1.0000e-06
Epoch 154/200
375/377 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - accuracy: 0.0695 - loss:
5.5084
Epoch 154: val_accuracy did not improve from 0.11296
377/377 ━━━━━━━━━━━━━━━━━━━━ 9s 23ms/step - accuracy: 0.0695 - loss:
5.5086 - val_accuracy: 0.1056 - val_loss: 5.1303 - learning_rate:
1.0000e-06
Epoch 155/200
375/377 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - accuracy: 0.0740 - loss:
5.3326
Epoch 155: val_accuracy did not improve from 0.11296
377/377 ━━━━━━━━━━━━━━━━━━━━ 9s 23ms/step - accuracy: 0.0740 - loss:
5.3338 - val_accuracy: 0.1017 - val_loss: 5.1270 - learning_rate:
1.0000e-06
Epoch 156/200
375/377 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - accuracy: 0.0729 - loss:
5.3617
Epoch 156: val_accuracy did not improve from 0.11296
377/377 ━━━━━━━━━━━━━━━━━━━━ 9s 23ms/step - accuracy: 0.0729 - loss:
5.3622 - val_accuracy: 0.1050 - val_loss: 5.1271 - learning_rate:
1.0000e-06
Epoch 157/200
375/377 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - accuracy: 0.0689 - loss:
5.4216
Epoch 157: val_accuracy did not improve from 0.11296
377/377 ━━━━━━━━━━━━━━━━━━━━ 9s 23ms/step - accuracy: 0.0689 - loss:
```

```
5.4221 - val_accuracy: 0.1070 - val_loss: 5.1133 - learning_rate:
1.0000e-06
Epoch 158/200
376/377 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - accuracy: 0.0625 - loss:
5.3595
Epoch 158: val_accuracy did not improve from 0.11296
377/377 ━━━━━━━━━━━━━━━━━━━━ 9s 24ms/step - accuracy: 0.0625 - loss:
5.3600 - val_accuracy: 0.1050 - val_loss: 5.1199 - learning_rate:
1.0000e-06
Epoch 159/200
375/377 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - accuracy: 0.0665 - loss:
5.5125
Epoch 159: val_accuracy did not improve from 0.11296
377/377 ━━━━━━━━━━━━━━━━━━━━ 9s 23ms/step - accuracy: 0.0665 - loss:
5.5123 - val_accuracy: 0.1070 - val_loss: 5.1243 - learning_rate:
1.0000e-06
Epoch 160/200
376/377 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - accuracy: 0.0757 - loss:
5.5629
Epoch 160: val_accuracy did not improve from 0.11296
377/377 ━━━━━━━━━━━━━━━━━━━━ 9s 23ms/step - accuracy: 0.0757 - loss:
5.5628 - val_accuracy: 0.1017 - val_loss: 5.1186 - learning_rate:
1.0000e-06
Epoch 161/200
376/377 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - accuracy: 0.0698 - loss:
5.3692
Epoch 161: val_accuracy did not improve from 0.11296
377/377 ━━━━━━━━━━━━━━━━━━━━ 9s 23ms/step - accuracy: 0.0698 - loss:
5.3695 - val_accuracy: 0.1037 - val_loss: 5.1216 - learning_rate:
1.0000e-06
Epoch 162/200
375/377 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - accuracy: 0.0760 - loss:
5.7339
Epoch 162: val_accuracy did not improve from 0.11296
377/377 ━━━━━━━━━━━━━━━━━━━━ 9s 24ms/step - accuracy: 0.0760 - loss:
5.7318 - val_accuracy: 0.1063 - val_loss: 5.1093 - learning_rate:
1.0000e-06
Epoch 163/200
375/377 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - accuracy: 0.0680 - loss:
5.5037
Epoch 163: val_accuracy did not improve from 0.11296
377/377 ━━━━━━━━━━━━━━━━━━━━ 9s 23ms/step - accuracy: 0.0680 - loss:
5.5030 - val_accuracy: 0.1050 - val_loss: 5.1057 - learning_rate:
1.0000e-06
Epoch 164/200
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - accuracy: 0.0651 - loss:
5.5564
Epoch 164: val_accuracy did not improve from 0.11296
377/377 ━━━━━━━━━━━━━━━━━━━━ 9s 23ms/step - accuracy: 0.0651 - loss:
```

```
5.5563 - val_accuracy: 0.1063 - val_loss: 5.1040 - learning_rate:
1.0000e-06
Epoch 165/200
375/377 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - accuracy: 0.0688 - loss:
5.6939
Epoch 165: val_accuracy did not improve from 0.11296
377/377 ━━━━━━━━━━━━━━━━━━━━ 9s 24ms/step - accuracy: 0.0688 - loss:
5.6920 - val_accuracy: 0.1056 - val_loss: 5.1016 - learning_rate:
1.0000e-06
Epoch 166/200
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - accuracy: 0.0765 - loss:
5.5851
Epoch 166: val_accuracy did not improve from 0.11296
377/377 ━━━━━━━━━━━━━━━━━━━━ 9s 23ms/step - accuracy: 0.0765 - loss:
5.5846 - val_accuracy: 0.1030 - val_loss: 5.0959 - learning_rate:
1.0000e-06
Epoch 167/200
375/377 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - accuracy: 0.0716 - loss:
5.3890
Epoch 167: val_accuracy did not improve from 0.11296
377/377 ━━━━━━━━━━━━━━━━━━━━ 9s 24ms/step - accuracy: 0.0716 - loss:
5.3890 - val_accuracy: 0.0990 - val_loss: 5.1010 - learning_rate:
1.0000e-06
Epoch 168/200
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - accuracy: 0.0655 - loss:
5.3121
Epoch 168: val_accuracy did not improve from 0.11296
377/377 ━━━━━━━━━━━━━━━━━━━━ 9s 24ms/step - accuracy: 0.0655 - loss:
5.3123 - val_accuracy: 0.1083 - val_loss: 5.0885 - learning_rate:
1.0000e-06
Epoch 169/200
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - accuracy: 0.0657 - loss:
5.6014
Epoch 169: val_accuracy did not improve from 0.11296
377/377 ━━━━━━━━━━━━━━━━━━━━ 9s 24ms/step - accuracy: 0.0658 - loss:
5.6009 - val_accuracy: 0.1043 - val_loss: 5.0855 - learning_rate:
1.0000e-06
Epoch 170/200
376/377 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - accuracy: 0.0690 - loss:
5.2507
Epoch 170: val_accuracy did not improve from 0.11296
377/377 ━━━━━━━━━━━━━━━━━━━━ 9s 23ms/step - accuracy: 0.0690 - loss:
5.2516 - val_accuracy: 0.1050 - val_loss: 5.0861 - learning_rate:
1.0000e-06
Epoch 171/200
375/377 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - accuracy: 0.0717 - loss:
5.3714
Epoch 171: val_accuracy did not improve from 0.11296
377/377 ━━━━━━━━━━━━━━━━━━━━ 9s 23ms/step - accuracy: 0.0718 - loss:
```

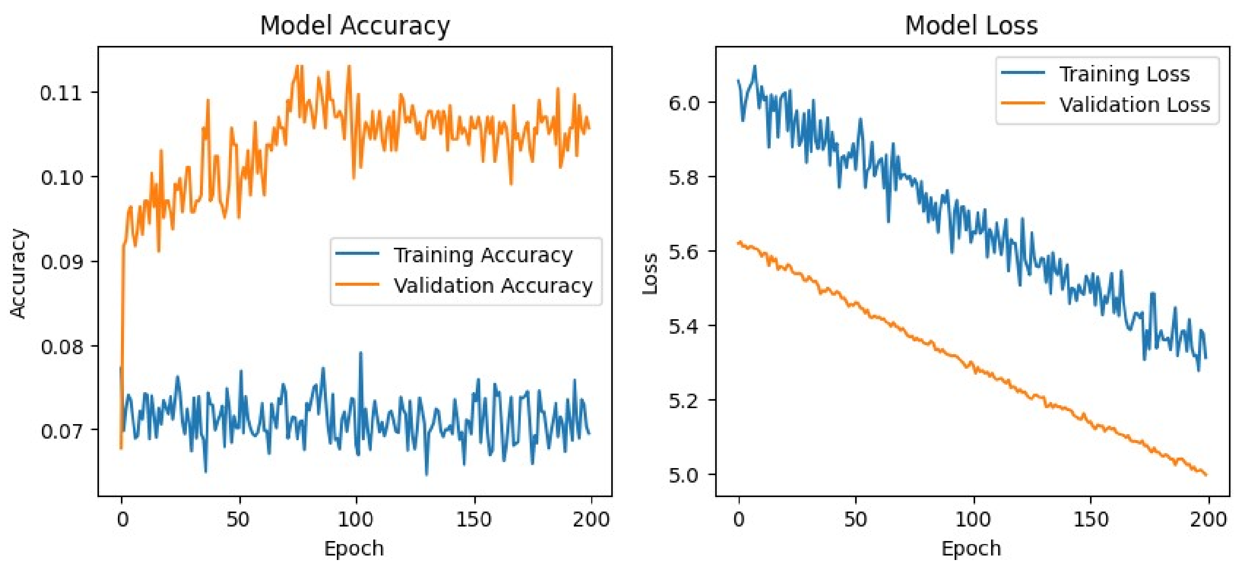
```
5.3719 - val_accuracy: 0.1030 - val_loss: 5.0841 - learning_rate:
1.0000e-06
Epoch 172/200
375/377 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - accuracy: 0.0757 - loss:
5.2303
Epoch 172: val_accuracy did not improve from 0.11296
377/377 ━━━━━━━━━━━━━━━━━━━━ 9s 23ms/step - accuracy: 0.0757 - loss:
5.2317 - val_accuracy: 0.1043 - val_loss: 5.0813 - learning_rate:
1.0000e-06
Epoch 173/200
376/377 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - accuracy: 0.0741 - loss:
5.4164
Epoch 173: val_accuracy did not improve from 0.11296
377/377 ━━━━━━━━━━━━━━━━━━━━ 9s 24ms/step - accuracy: 0.0741 - loss:
5.4164 - val_accuracy: 0.1056 - val_loss: 5.0877 - learning_rate:
1.0000e-06
Epoch 174/200
375/377 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - accuracy: 0.0772 - loss:
5.3449
Epoch 174: val_accuracy did not improve from 0.11296
377/377 ━━━━━━━━━━━━━━━━━━━━ 9s 23ms/step - accuracy: 0.0772 - loss:
5.3445 - val_accuracy: 0.1063 - val_loss: 5.0744 - learning_rate:
1.0000e-06
Epoch 175/200
376/377 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - accuracy: 0.0767 - loss:
5.4196
Epoch 175: val_accuracy did not improve from 0.11296
377/377 ━━━━━━━━━━━━━━━━━━━━ 9s 23ms/step - accuracy: 0.0767 - loss:
5.4194 - val_accuracy: 0.1030 - val_loss: 5.0750 - learning_rate:
1.0000e-06
Epoch 176/200
375/377 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - accuracy: 0.0681 - loss:
5.3765
Epoch 176: val_accuracy did not improve from 0.11296
377/377 ━━━━━━━━━━━━━━━━━━━━ 9s 24ms/step - accuracy: 0.0681 - loss:
5.3762 - val_accuracy: 0.1017 - val_loss: 5.0653 - learning_rate:
1.0000e-06
Epoch 177/200
376/377 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - accuracy: 0.0691 - loss:
5.7412
Epoch 177: val_accuracy did not improve from 0.11296
377/377 ━━━━━━━━━━━━━━━━━━━━ 9s 23ms/step - accuracy: 0.0691 - loss:
5.7399 - val_accuracy: 0.1056 - val_loss: 5.0560 - learning_rate:
1.0000e-06
Epoch 178/200
376/377 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - accuracy: 0.0642 - loss:
5.4525
Epoch 178: val_accuracy did not improve from 0.11296
377/377 ━━━━━━━━━━━━━━━━━━━━ 9s 23ms/step - accuracy: 0.0643 - loss:
```

```
5.4527 - val_accuracy: 0.1050 - val_loss: 5.0685 - learning_rate:
1.0000e-06
Epoch 179/200
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - accuracy: 0.0744 - loss:
5.2117
Epoch 179: val_accuracy did not improve from 0.11296
377/377 ━━━━━━━━━━━━━━━━━━━━ 9s 23ms/step - accuracy: 0.0744 - loss:
5.2120 - val_accuracy: 0.1090 - val_loss: 5.0564 - learning_rate:
1.0000e-06
Epoch 180/200
375/377 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - accuracy: 0.0752 - loss:
5.2938
Epoch 180: val_accuracy did not improve from 0.11296
377/377 ━━━━━━━━━━━━━━━━━━━━ 9s 24ms/step - accuracy: 0.0751 - loss:
5.2944 - val_accuracy: 0.1063 - val_loss: 5.0568 - learning_rate:
1.0000e-06
Epoch 181/200
375/377 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - accuracy: 0.0706 - loss:
5.3078
Epoch 181: val_accuracy did not improve from 0.11296
377/377 ━━━━━━━━━━━━━━━━━━━━ 9s 23ms/step - accuracy: 0.0707 - loss:
5.3084 - val_accuracy: 0.1070 - val_loss: 5.0479 - learning_rate:
1.0000e-06
Epoch 182/200
375/377 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - accuracy: 0.0695 - loss:
5.3417
Epoch 182: val_accuracy did not improve from 0.11296
377/377 ━━━━━━━━━━━━━━━━━━━━ 9s 23ms/step - accuracy: 0.0695 - loss:
5.3418 - val_accuracy: 0.1070 - val_loss: 5.0453 - learning_rate:
1.0000e-06
Epoch 183/200
376/377 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - accuracy: 0.0649 - loss:
5.4356
Epoch 183: val_accuracy did not improve from 0.11296
377/377 ━━━━━━━━━━━━━━━━━━━━ 9s 24ms/step - accuracy: 0.0650 - loss:
5.4352 - val_accuracy: 0.1050 - val_loss: 5.0496 - learning_rate:
1.0000e-06
Epoch 184/200
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - accuracy: 0.0685 - loss:
5.1012
Epoch 184: val_accuracy did not improve from 0.11296
377/377 ━━━━━━━━━━━━━━━━━━━━ 9s 23ms/step - accuracy: 0.0685 - loss:
5.1019 - val_accuracy: 0.1056 - val_loss: 5.0456 - learning_rate:
1.0000e-06
Epoch 185/200
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - accuracy: 0.0757 - loss:
5.0535
Epoch 185: val_accuracy did not improve from 0.11296
377/377 ━━━━━━━━━━━━━━━━━━━━ 9s 23ms/step - accuracy: 0.0757 - loss:
```

```
5.0542 - val_accuracy: 0.1070 - val_loss: 5.0377 - learning_rate:
1.0000e-06
Epoch 186/200
376/377 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - accuracy: 0.0734 - loss:
5.5977
Epoch 186: val_accuracy did not improve from 0.11296
377/377 ━━━━━━━━━━━━━━━━━━━━ 9s 23ms/step - accuracy: 0.0734 - loss:
5.5966 - val_accuracy: 0.1037 - val_loss: 5.0378 - learning_rate:
1.0000e-06
Epoch 187/200
375/377 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - accuracy: 0.0733 - loss:
5.3905
Epoch 187: val_accuracy did not improve from 0.11296
377/377 ━━━━━━━━━━━━━━━━━━━━ 9s 24ms/step - accuracy: 0.0733 - loss:
5.3909 - val_accuracy: 0.1103 - val_loss: 5.0228 - learning_rate:
1.0000e-06
Epoch 188/200
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - accuracy: 0.0754 - loss:
5.0665
Epoch 188: val_accuracy did not improve from 0.11296
377/377 ━━━━━━━━━━━━━━━━━━━━ 9s 23ms/step - accuracy: 0.0754 - loss:
5.0672 - val_accuracy: 0.1010 - val_loss: 5.0375 - learning_rate:
1.0000e-06
Epoch 189/200
375/377 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - accuracy: 0.0662 - loss:
5.2825
Epoch 189: val_accuracy did not improve from 0.11296
377/377 ━━━━━━━━━━━━━━━━━━━━ 9s 23ms/step - accuracy: 0.0663 - loss:
5.2830 - val_accuracy: 0.1023 - val_loss: 5.0381 - learning_rate:
1.0000e-06
Epoch 190/200
375/377 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - accuracy: 0.0742 - loss:
5.2406
Epoch 190: val_accuracy did not improve from 0.11296
377/377 ━━━━━━━━━━━━━━━━━━━━ 9s 23ms/step - accuracy: 0.0741 - loss:
5.2418 - val_accuracy: 0.1050 - val_loss: 5.0384 - learning_rate:
1.0000e-06
Epoch 191/200
376/377 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - accuracy: 0.0729 - loss:
5.1897
Epoch 191: val_accuracy did not improve from 0.11296
377/377 ━━━━━━━━━━━━━━━━━━━━ 9s 24ms/step - accuracy: 0.0729 - loss:
5.1907 - val_accuracy: 0.1030 - val_loss: 5.0307 - learning_rate:
1.0000e-06
Epoch 192/200
375/377 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - accuracy: 0.0692 - loss:
4.9980
Epoch 192: val_accuracy did not improve from 0.11296
377/377 ━━━━━━━━━━━━━━━━━━━━ 9s 23ms/step - accuracy: 0.0693 - loss:
```

```
5.0007 - val_accuracy: 0.1056 - val_loss: 5.0233 - learning_rate:
1.0000e-06
Epoch 193/200
375/377 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - accuracy: 0.0739 - loss:
5.6669
Epoch 193: val_accuracy did not improve from 0.11296
377/377 ━━━━━━━━━━━━━━━━━━━━ 9s 23ms/step - accuracy: 0.0738 - loss:
5.6649 - val_accuracy: 0.1056 - val_loss: 5.0245 - learning_rate:
1.0000e-06
Epoch 194/200
375/377 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - accuracy: 0.0765 - loss:
5.4763
Epoch 194: val_accuracy did not improve from 0.11296
377/377 ━━━━━━━━━━━━━━━━━━━━ 9s 24ms/step - accuracy: 0.0765 - loss:
5.4752 - val_accuracy: 0.1096 - val_loss: 5.0110 - learning_rate:
1.0000e-06
Epoch 195/200
376/377 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - accuracy: 0.0678 - loss:
5.3697
Epoch 195: val_accuracy did not improve from 0.11296
377/377 ━━━━━━━━━━━━━━━━━━━━ 9s 23ms/step - accuracy: 0.0678 - loss:
5.3694 - val_accuracy: 0.1023 - val_loss: 5.0171 - learning_rate:
1.0000e-06
Epoch 196/200
375/377 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - accuracy: 0.0744 - loss:
5.4133
Epoch 196: val_accuracy did not improve from 0.11296
377/377 ━━━━━━━━━━━━━━━━━━━━ 9s 23ms/step - accuracy: 0.0743 - loss:
5.4126 - val_accuracy: 0.1083 - val_loss: 5.0049 - learning_rate:
1.0000e-06
Epoch 197/200
375/377 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - accuracy: 0.0743 - loss:
5.2089
Epoch 197: val_accuracy did not improve from 0.11296
377/377 ━━━━━━━━━━━━━━━━━━━━ 9s 23ms/step - accuracy: 0.0743 - loss:
5.2095 - val_accuracy: 0.1056 - val_loss: 5.0066 - learning_rate:
1.0000e-06
Epoch 198/200
377/377 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - accuracy: 0.0716 - loss:
5.4750
Epoch 198: val_accuracy did not improve from 0.11296
377/377 ━━━━━━━━━━━━━━━━━━━━ 9s 24ms/step - accuracy: 0.0716 - loss:
5.4747 - val_accuracy: 0.1050 - val_loss: 5.0095 - learning_rate:
1.0000e-06
Epoch 199/200
375/377 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - accuracy: 0.0716 - loss:
5.2169
Epoch 199: val_accuracy did not improve from 0.11296
377/377 ━━━━━━━━━━━━━━━━━━━━ 9s 23ms/step - accuracy: 0.0716 - loss:
```

```
5.2182 - val_accuracy: 0.1070 - val_loss: 5.0025 - learning_rate:
1.0000e-06
Epoch 200/200
375/377 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - accuracy: 0.0693 - loss:
5.2926
Epoch 200: val_accuracy did not improve from 0.11296
377/377 ━━━━━━━━━━━━━━━━━━━━ 9s 23ms/step - accuracy: 0.0693 - loss:
5.2928 - val_accuracy: 0.1056 - val_loss: 4.9957 - learning_rate:
1.0000e-06

# Training perfermance report

plt.rcParams['figure.figsize'] = (10, 4)

# Plotting accuracy
plt.subplot(1, 2, 1)
plt.plot(history.history['accuracy'], label='Training Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.title('Model Accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.legend()

# Plotting loss
plt.subplot(1, 2, 2)
plt.plot(history.history['loss'], label='Training Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.title('Model Loss')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend()
plt.show()
```



```
# Saving the model

model.save('final_model.h5')


# Load the model from the file

model = load_model('/kaggle/working/final_model.h5')

# Testing the custom model's accuracy
test_loss, test_accuracy = model.evaluate(padded_test_sequences,
y_test, verbose=1)
print(f'Test Accuracy: {(test_accuracy * 100):.2f}%')


48/48 ━━━━━━━━━━━━━━━━━━━━ 1s 8ms/step - accuracy: 0.1033 - loss:
4.9594
Test Accuracy: 11.02%

# Plotting the confusion matrix

y_pred = np.argmax(model.predict(padded_test_sequences), axis=-1) #
Predict the labels for test data
y_true = np.argmax(y_test, axis=-1)

class_names = label_encoder.classes_

cm = confusion_matrix(y_true, y_pred) # Confusion matrix
# cm = cm.astype("float") / cm.sum(axis=1)[:, np.newaxis]
plt.figure(figsize=(10, 10))
sns.heatmap(cm, annot=True, cmap="Blues", fmt="d",
xticklabels=class_names, yticklabels=class_names)
plt.title("Confusion Matrix")
plt.ylabel("True Label")
plt.xlabel("Predicted Label")
plt.savefig('cm_cf.png')
plt.show()


48/48 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step

----------------------------------------------------------------------
-----
TypeError                                 Traceback (most recent call
last)
<ipython-input-166-5ff472417548> in <cell line: 8>()
      6 class_names = label_encoder.classes_
      7
----> 8 cm = confusion_matrix(y_true, y_pred) # Confusion matrix
      9 # cm = cm.astype("float") / cm.sum(axis=1)[:, np.newaxis]
     10 plt.figure(figsize=(10, 10))
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classificatio
n.py in confusion_matrix(y_true, y_pred, labels, sample_weight,
normalize)
    315     (0, 2, 1, 1)
    316     """
--> 317     y_type, y_true, y_pred = _check_targets(y_true, y_pred)
    318     if y_type not in ("binary", "multiclass"):
    319         raise ValueError("%s is not supported" % y_type)

/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classificatio
n.py in _check_targets(y_true, y_pred)
     84     y_pred : array or indicator matrix
     85     """
---> 86     check_consistent_length(y_true, y_pred)
     87     type_true = type_of_target(y_true, input_name="y_true")
     88     type_pred = type_of_target(y_pred, input_name="y_pred")

/usr/local/lib/python3.10/dist-packages/sklearn/utils/validation.py in
check_consistent_length(*arrays)
    392     """
    393
--> 394     lengths = [_num_samples(X) for X in arrays if X is not
None]
    395     uniques = np.unique(lengths)
    396     if len(uniques) > 1:

/usr/local/lib/python3.10/dist-packages/sklearn/utils/validation.py in
<listcomp>(.0)
    392     """
    393
--> 394     lengths = [_num_samples(X) for X in arrays if X is not
None]
    395     uniques = np.unique(lengths)
    396     if len(uniques) > 1:

/usr/local/lib/python3.10/dist-packages/sklearn/utils/validation.py in
_num_samples(x)
    333     if hasattr(x, "shape") and x.shape is not None:
    334         if len(x.shape) == 0:
--> 335             raise TypeError(
    336                 "Singleton array %r cannot be considered a
valid collection." % x
    337             )

TypeError: Singleton array 934 cannot be considered a valid
collection.

# Classification report
print(classification_report(y_true, y_pred, target_names=class_names))
```

	precision	recall	f1-score	support
Action	0.50	0.00	0.01	243
Adventure	0.05	0.16	0.07	63
Animation	0.04	0.07	0.05	60
Biography	0.11	0.30	0.16	83
Comedy	0.32	0.24	0.27	382
Crime	0.08	0.06	0.07	120
Documentary	0.12	0.13	0.13	135
Drama	0.28	0.01	0.03	346
Family	0.00	0.00	0.00	1
Fantasy	0.01	0.50	0.02	6
Game-Show	0.12	0.50	0.20	2
Horror	0.06	0.02	0.03	52
Mystery	0.00	0.00	0.00	3
Reality-TV	0.00	0.00	0.00	5
Romance	0.00	0.00	0.00	1
Sci-Fi	0.00	0.00	0.00	1
Thriller	0.00	0.00	0.00	3
accuracy			0.11	1506
macro avg	0.10	0.12	0.06	1506
weighted avg	0.26	0.11	0.11	1506