Using minimum support = 0.01 and minimum confidence threshold = 0.1, what are the association rules you can extract from your dataset? (0.5 point)

In [6]:
```
import pandas as pd
from mlxtend.preprocessing import TransactionEncoder
from mlxtend.frequent_patterns import fpgrowth
from mlxtend.frequent_patterns import association_rules

dataset_url = '/Users/sarahsamhithachella/Downloads/Grocery_Items_10.csv'

df = pd.read_csv('/Users/sarahsamhithachella/Downloads/Grocery_Items_10.csv')

transactions = df.apply(lambda row: row.dropna().tolist(), axis=1).tolist()
te = TransactionEncoder()
te_ary = te.fit(transactions).transform(transactions)

df_transformed = pd.DataFrame(te_ary, columns=te.columns_)

frequent_itemsets = fpgrowth(df_transformed, min_support=0.01, use_colnames=True)
rules = association_rules(frequent_itemsets, metric="confidence", min_threshold=0.1)

print("Frequent Itemsets:")
print(frequent_itemsets)

print("\nAssociation Rules:")
rules.to_csv('association_rules.csv', index=False)
display(rules)
```

```
Frequent Itemsets:
     support                     itemsets
0   0.018750                      (sugar)
1   0.015625                  (beverages)
2   0.058375               (bottled water)
3   0.037875                     (butter)
4   0.027125                    (chicken)
..       ...                          ...
62  0.014375         (whole milk, rolls/buns)
63  0.010750  (other vegetables, rolls/buns)
64  0.016000  (other vegetables, whole milk)
65  0.012125             (whole milk, yogurt)
66  0.012125               (soda, whole milk)

[67 rows x 2 columns]

Association Rules:
```

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage	conviction	zhangs_metric
0	(rolls/buns)	(whole milk)	0.111625	0.15750	0.014375	0.128779	0.817647	-0.003206	0.967034	-0.200668
1	(other vegetables)	(whole milk)	0.119750	0.15750	0.016000	0.133612	0.848328	-0.002861	0.972428	-0.168822
2	(whole milk)	(other vegetables)	0.157500	0.11975	0.016000	0.101587	0.848328	-0.002861	0.979784	-0.175062
3	(yogurt)	(whole milk)	0.088750	0.15750	0.012125	0.136620	0.867427	-0.001853	0.975816	-0.143630
4	(soda)	(whole milk)	0.098625	0.15750	0.012125	0.122940	0.780574	-0.003408	0.960596	-0.237727

Use minimum support values (msv): 0.001, 0.005, 0.01, 0.05 and minimum confidence threshold (mct): 0.05, 0.075, 0.1. For each pair (msv, mct), find the number of association rules extracted from the dataset. Construct a heatmap using Seaborn data visualization library to show the count results such that the x- axis is msv and the y-axis is mct. (2.5 points)

In [18]:

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from mlxtend.preprocessing import TransactionEncoder
from mlxtend.frequent_patterns import fpgrowth, association_rules

df = pd.read_csv('/Users/sarahsamhithachella/Downloads/Grocery_Items_10.csv')

transactions = df.apply(lambda row: row.dropna().tolist(), axis=1).tolist()
te = TransactionEncoder()
te_ary = te.fit(transactions).transform(transactions)

df_transformed = pd.DataFrame(te_ary, columns=te.columns_)

msv_values = [0.001, 0.005, 0.01, 0.05]
mct_values = [0.05, 0.075, 0.1]

rules_counts = pd.DataFrame(index=msv_values, columns=mct_values)

for msv in msv_values:
    for mct in mct_values:
        frequent_itemsets = fpgrowth(df_transformed, min_support=msv, use_colnames=True)

        rules = association_rules(frequent_itemsets, metric="confidence", min_threshold=mct)

        rules_counts.at[msv, mct] = len(rules)

        print(f"Number of rules for (msv={msv}, mct={mct}): {len(rules)}")

rules_counts = rules_counts.apply(pd.to_numeric)
```

```python
plt.figure(figsize=(10, 6))
sns.heatmap(rules_counts, annot=True, cmap="Blues",fmt="g", linewidths=.5)
plt.title("Number of Association Rules")
plt.xlabel("Minimum Confidence Threshold (mct)")
plt.ylabel("Minimum Support Value (msv)")
plt.show()
```

```
Number of rules for (msv=0.001, mct=0.05): 534
Number of rules for (msv=0.001, mct=0.075): 307
Number of rules for (msv=0.001, mct=0.1): 162
Number of rules for (msv=0.005, mct=0.05): 61
Number of rules for (msv=0.005, mct=0.075): 42
Number of rules for (msv=0.005, mct=0.1): 23
Number of rules for (msv=0.01, mct=0.05): 10
Number of rules for (msv=0.01, mct=0.075): 10
Number of rules for (msv=0.01, mct=0.1): 5
Number of rules for (msv=0.05, mct=0.05): 0
Number of rules for (msv=0.05, mct=0.075): 0
Number of rules for (msv=0.05, mct=0.1): 0
```

List the association rule(s) (i.e., one or more rules depending on your dataset) that have the highest confidence for minimum support = 0.005. What is that confidence value? (1 point)

In [8]:

```
import pandas as pd
from mlxtend.preprocessing import TransactionEncoder
from mlxtend.frequent_patterns import fpgrowth, association_rules

df = pd.read_csv('/Users/sarahsamhithachella/Downloads/Grocery_Items_10.csv')

transactions = df.apply(lambda row: row.dropna().tolist(), axis=1).tolist()
te = TransactionEncoder()
te_ary = te.fit(transactions).transform(transactions)
```

```python
df_transformed = pd.DataFrame(te_ary, columns=te.columns_)

min_support = 0.005

frequent_itemsets = fpgrowth(df_transformed, min_support=min_support, use_colnames=True)

rules = association_rules(frequent_itemsets, metric="confidence", min_threshold=0.0)

max_confidence_rule = rules.loc[rules['confidence'].idxmax()]

print("Association Rule(s) with the Highest Confidence:")
print(max_confidence_rule)
```

```
Association Rule(s) with the Highest Confidence:
antecedents            (bottled beer)
consequents              (whole milk)
antecedent support           0.042125
consequent support             0.1575
support                      0.008125
confidence                   0.192878
lift                         1.224624
leverage                      0.00149
conviction                   1.043833
zhangs_metric                 0.19149
Name: 49, dtype: object
```

High Confidence Value

The confidence value is 0.192878. A confidence level of 0.192878 indicates that "whole milk" is purchased in almost 19.29% of the cases in which "bottled beer" is present.

In [68]:

```
pip install tensorflow
```

```
Collecting tensorflow
  Obtaining dependency information for tensorflow from https://files.pythonhosted.org/packages/85/15/cf99a373812d37f8ae99752a34
a9f5f690d820ceb5b302e922705bc18944/tensorflow-2.15.0-cp311-cp311-macosx_12_0_arm64.whl.metadata
  Downloading tensorflow-2.15.0-cp311-cp311-macosx_12_0_arm64.whl.metadata (3.6 kB)
Collecting tensorflow-macos==2.15.0 (from tensorflow)
  Obtaining dependency information for tensorflow-macos==2.15.0 from https://files.pythonhosted.org/packages/eb/9f/0759e2fea4a3
c48f070b64811c2c57036b46353ba87263afc810b8f4188a/tensorflow_macos-2.15.0-cp311-cp311-macosx_12_0_arm64.whl.metadata
  Downloading tensorflow_macos-2.15.0-cp311-cp311-macosx_12_0_arm64.whl.metadata (4.2 kB)
Collecting absl-py>=1.0.0 (from tensorflow-macos==2.15.0->tensorflow)
  Obtaining dependency information for absl-py>=1.0.0 from https://files.pythonhosted.org/packages/01/e4/dc0a1dcc4e74e08d7abeda
b278c795eef54a224363bb18f5692f416d834f/absl_py-2.0.0-py3-none-any.whl.metadata
  Downloading absl_py-2.0.0-py3-none-any.whl.metadata (2.3 kB)
Collecting astunparse>=1.6.0 (from tensorflow-macos==2.15.0->tensorflow)
  Downloading astunparse-1.6.3-py2.py3-none-any.whl (12 kB)
Collecting flatbuffers>=23.5.26 (from tensorflow-macos==2.15.0->tensorflow)
  Obtaining dependency information for flatbuffers>=23.5.26 from https://files.pythonhosted.org/packages/6f/12/d5c79ee252793ffe
845d58a913197bfa02ae9a0b5c9bc3dc4b58d477b9e7/flatbuffers-23.5.26-py2.py3-none-any.whl.metadata
  Downloading flatbuffers-23.5.26-py2.py3-none-any.whl.metadata (850 bytes)
Collecting gast!=0.5.0,!=0.5.1,!=0.5.2,>=0.2.1 (from tensorflow-macos==2.15.0->tensorflow)
  Downloading gast-0.5.4-py3-none-any.whl (19 kB)
Collecting google-pasta>=0.1.1 (from tensorflow-macos==2.15.0->tensorflow)
  Downloading google_pasta-0.2.0-py3-none-any.whl (57 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 57.5/57.5 kB 2.5 MB/s eta 0:00:00
Requirement already satisfied: h5py>=2.9.0 in /Users/sarahsamhithachella/anaconda3/lib/python3.11/site-packages (from tensorflo
w-macos==2.15.0->tensorflow) (3.7.0)
Collecting libclang>=13.0.0 (from tensorflow-macos==2.15.0->tensorflow)
  Obtaining dependency information for libclang>=13.0.0 from https://files.pythonhosted.org/packages/32/1f/981809b77b71972beec3
4b3ff5422c1b1f7e519daac7b3cbd055c05ba2cf/libclang-16.0.6-py2.py3-none-macosx_11_0_arm64.whl.metadata
  Downloading libclang-16.0.6-py2.py3-none-macosx_11_0_arm64.whl.metadata (5.2 kB)
Collecting ml-dtypes~=0.2.0 (from tensorflow-macos==2.15.0->tensorflow)
  Obtaining dependency information for ml-dtypes~=0.2.0 from https://files.pythonhosted.org/packages/15/da/43bee505963da0c730ee
50e951c604bfdb90d4cccc9c0044c946b10e68a7/ml_dtypes-0.2.0-cp311-cp311-macosx_10_9_universal2.whl.metadata
  Downloading ml_dtypes-0.2.0-cp311-cp311-macosx_10_9_universal2.whl.metadata (20 kB)
Requirement already satisfied: numpy<2.0.0,>=1.23.5 in /Users/sarahsamhithachella/anaconda3/lib/python3.11/site-packages (from
tensorflow-macos==2.15.0->tensorflow) (1.24.3)
Collecting opt-einsum>=2.3.2 (from tensorflow-macos==2.15.0->tensorflow)
  Downloading opt_einsum-3.3.0-py3-none-any.whl (65 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 65.5/65.5 kB 6.9 MB/s eta 0:00:00
Requirement already satisfied: packaging in /Users/sarahsamhithachella/anaconda3/lib/python3.11/site-packages (from tensorflow-
macos==2.15.0->tensorflow) (23.0)
Collecting protobuf!=4.21.0,!=4.21.1,!=4.21.2,!=4.21.3,!=4.21.4,!=4.21.5,<5.0.0dev,>=3.20.3 (from tensorflow-macos==2.15.0->ten
sorflow)
  Obtaining dependency information for protobuf!=4.21.0,!=4.21.1,!=4.21.2,!=4.21.3,!=4.21.4,!=4.21.5,<5.0.0dev,>=3.20.3 from ht
tps://files.pythonhosted.org/packages/e6/db/7b2edc72807d45d72f9db42f3eb86ddaf37f9e55d923159b1dbfc9d835bc/protobuf-4.25.1-cp37-a
bi3-macosx_10_9_universal2.whl.metadata
  Downloading protobuf-4.25.1-cp37-abi3-macosx_10_9_universal2.whl.metadata (541 bytes)
Requirement already satisfied: setuptools in /Users/sarahsamhithachella/anaconda3/lib/python3.11/site-packages (from tensorflow
-macos==2.15.0->tensorflow) (68.0.0)
```

```
Requirement already satisfied: six>=1.12.0 in /Users/sarahsamhithachella/anaconda3/lib/python3.11/site-packages (from tensorflo
w-macos==2.15.0->tensorflow) (1.16.0)
Collecting termcolor>=1.1.0 (from tensorflow-macos==2.15.0->tensorflow)
  Downloading termcolor-2.3.0-py3-none-any.whl (6.9 kB)
Requirement already satisfied: typing-extensions>=3.6.6 in /Users/sarahsamhithachella/anaconda3/lib/python3.11/site-packages (f
rom tensorflow-macos==2.15.0->tensorflow) (4.7.1)
Requirement already satisfied: wrapt<1.15,>=1.11.0 in /Users/sarahsamhithachella/anaconda3/lib/python3.11/site-packages (from t
ensorflow-macos==2.15.0->tensorflow) (1.14.1)
Collecting tensorflow-io-gcs-filesystem>=0.23.1 (from tensorflow-macos==2.15.0->tensorflow)
  Obtaining dependency information for tensorflow-io-gcs-filesystem>=0.23.1 from https://files.pythonhosted.org/packages/5b/e9/
1444afc87596a90066704cc46ed661a4e7b348eec03a3fc2ca10ab917254/tensorflow_io_gcs_filesystem-0.34.0-cp311-cp311-macosx_12_0_arm64.
whl.metadata
  Downloading tensorflow_io_gcs_filesystem-0.34.0-cp311-cp311-macosx_12_0_arm64.whl.metadata (14 kB)
Collecting grpcio<2.0,>=1.24.3 (from tensorflow-macos==2.15.0->tensorflow)
  Obtaining dependency information for grpcio<2.0,>=1.24.3 from https://files.pythonhosted.org/packages/92/93/3cbc00a269b46277f
f26355074a8315eeb4c87240c27d6f7efeabe818fd9/grpcio-1.59.3-cp311-cp311-macosx_10_10_universal2.whl.metadata
  Downloading grpcio-1.59.3-cp311-cp311-macosx_10_10_universal2.whl.metadata (4.0 kB)
Collecting tensorboard<2.16,>=2.15 (from tensorflow-macos==2.15.0->tensorflow)
  Obtaining dependency information for tensorboard<2.16,>=2.15 from https://files.pythonhosted.org/packages/6e/0c/1059a6682cf2c
c1fcc0d5327837b5672fe4f5574255fa5430d0a8ceb75e9/tensorboard-2.15.1-py3-none-any.whl.metadata
  Downloading tensorboard-2.15.1-py3-none-any.whl.metadata (1.7 kB)
Collecting tensorflow-estimator<2.16,>=2.15.0 (from tensorflow-macos==2.15.0->tensorflow)
  Obtaining dependency information for tensorflow-estimator<2.16,>=2.15.0 from https://files.pythonhosted.org/packages/b6/c8/2f
823c8958d5342eafc6dd3e922f0cc4fcf8c2e0460284cc462dae3b60a0/tensorflow_estimator-2.15.0-py2.py3-none-any.whl.metadata
  Downloading tensorflow_estimator-2.15.0-py2.py3-none-any.whl.metadata (1.3 kB)
Collecting keras<2.16,>=2.15.0 (from tensorflow-macos==2.15.0->tensorflow)
  Obtaining dependency information for keras<2.16,>=2.15.0 from https://files.pythonhosted.org/packages/fc/a7/0d4490de967a67f68
a538cc9cdb259bff971c4b5787f7765dc7c8f118f71/keras-2.15.0-py3-none-any.whl.metadata
  Downloading keras-2.15.0-py3-none-any.whl.metadata (2.4 kB)
Requirement already satisfied: wheel<1.0,>=0.23.0 in /Users/sarahsamhithachella/anaconda3/lib/python3.11/site-packages (from as
tunparse>=1.6.0->tensorflow-macos==2.15.0->tensorflow) (0.38.4)
Collecting google-auth<3,>=1.6.3 (from tensorboard<2.16,>=2.15->tensorflow-macos==2.15.0->tensorflow)
  Obtaining dependency information for google-auth<3,>=1.6.3 from https://files.pythonhosted.org/packages/86/a7/75911c13a242735
d5aeaca6a272da380335ff4ba5f26d6b2ae20ff682d13/google_auth-2.23.4-py2.py3-none-any.whl.metadata
  Downloading google_auth-2.23.4-py2.py3-none-any.whl.metadata (4.7 kB)
Collecting google-auth-oauthlib<2,>=0.5 (from tensorboard<2.16,>=2.15->tensorflow-macos==2.15.0->tensorflow)
  Obtaining dependency information for google-auth-oauthlib<2,>=0.5 from https://files.pythonhosted.org/packages/ce/33/a907b4b6
7245647746dde8d61e1643ef5d210c88e090d491efd89eff9f95/google_auth_oauthlib-1.1.0-py2.py3-none-any.whl.metadata
  Downloading google_auth_oauthlib-1.1.0-py2.py3-none-any.whl.metadata (2.7 kB)
Requirement already satisfied: markdown>=2.6.8 in /Users/sarahsamhithachella/anaconda3/lib/python3.11/site-packages (from tenso
rboard<2.16,>=2.15->tensorflow-macos==2.15.0->tensorflow) (3.4.1)
Collecting protobuf!=4.21.0,!=4.21.1,!=4.21.2,!=4.21.3,!=4.21.4,!=4.21.5,<5.0.0dev,>=3.20.3 (from tensorflow-macos==2.15.0->ten
sorflow)
  Obtaining dependency information for protobuf!=4.21.0,!=4.21.1,!=4.21.2,!=4.21.3,!=4.21.4,!=4.21.5,<5.0.0dev,>=3.20.3 from ht
tps://files.pythonhosted.org/packages/cb/d3/a164038605494d49acc4f9cda1c0bc200b96382c53edd561387263bb181d/protobuf-4.23.4-cp37-a
bi3-macosx_10_9_universal2.whl.metadata
  Downloading protobuf-4.23.4-cp37-abi3-macosx_10_9_universal2.whl.metadata (540 bytes)
Requirement already satisfied: requests<3,>=2.21.0 in /Users/sarahsamhithachella/anaconda3/lib/python3.11/site-packages (from t
```

```
ensorboard<2.16,>=2.15->tensorflow-macos==2.15.0->tensorflow) (2.31.0)
Collecting tensorboard-data-server<0.8.0,>=0.7.0 (from tensorboard<2.16,>=2.15->tensorflow-macos==2.15.0->tensorflow)
  Obtaining dependency information for tensorboard-data-server<0.8.0,>=0.7.0 from https://files.pythonhosted.org/packages/7a/1
3/e503968fefabd4c6b2650af21e110aa8466fe21432cd7c43a84577a89438/tensorboard_data_server-0.7.2-py3-none-any.whl.metadata
  Downloading tensorboard_data_server-0.7.2-py3-none-any.whl.metadata (1.1 kB)
Requirement already satisfied: werkzeug>=1.0.1 in /Users/sarahsamhithachella/anaconda3/lib/python3.11/site-packages (from tenso
rboard<2.16,>=2.15->tensorflow-macos==2.15.0->tensorflow) (2.2.3)
Collecting cachetools<6.0,>=2.0.0 (from google-auth<3,>=1.6.3->tensorboard<2.16,>=2.15->tensorflow-macos==2.15.0->tensorflow)
  Obtaining dependency information for cachetools<6.0,>=2.0.0 from https://files.pythonhosted.org/packages/a2/91/2d843adb9fbd91
1e0da45fbf6f18ca89d07a087c3daa23e955584f90ebf4/cachetools-5.3.2-py3-none-any.whl.metadata
  Downloading cachetools-5.3.2-py3-none-any.whl.metadata (5.2 kB)
Requirement already satisfied: pyasn1-modules>=0.2.1 in /Users/sarahsamhithachella/anaconda3/lib/python3.11/site-packages (from
google-auth<3,>=1.6.3->tensorboard<2.16,>=2.15->tensorflow-macos==2.15.0->tensorflow) (0.2.8)
Collecting rsa<5,>=3.1.4 (from google-auth<3,>=1.6.3->tensorboard<2.16,>=2.15->tensorflow-macos==2.15.0->tensorflow)
  Downloading rsa-4.9-py3-none-any.whl (34 kB)
Collecting requests-oauthlib>=0.7.0 (from google-auth-oauthlib<2,>=0.5->tensorboard<2.16,>=2.15->tensorflow-macos==2.15.0->tens
orflow)
  Downloading requests_oauthlib-1.3.1-py2.py3-none-any.whl (23 kB)
Requirement already satisfied: charset-normalizer<4,>=2 in /Users/sarahsamhithachella/anaconda3/lib/python3.11/site-packages (f
rom requests<3,>=2.21.0->tensorboard<2.16,>=2.15->tensorflow-macos==2.15.0->tensorflow) (2.0.4)
Requirement already satisfied: idna<4,>=2.5 in /Users/sarahsamhithachella/anaconda3/lib/python3.11/site-packages (from requests
<3,>=2.21.0->tensorboard<2.16,>=2.15->tensorflow-macos==2.15.0->tensorflow) (3.4)
Requirement already satisfied: urllib3<3,>=1.21.1 in /Users/sarahsamhithachella/anaconda3/lib/python3.11/site-packages (from re
quests<3,>=2.21.0->tensorboard<2.16,>=2.15->tensorflow-macos==2.15.0->tensorflow) (1.26.16)
Requirement already satisfied: certifi>=2017.4.17 in /Users/sarahsamhithachella/anaconda3/lib/python3.11/site-packages (from re
quests<3,>=2.21.0->tensorboard<2.16,>=2.15->tensorflow-macos==2.15.0->tensorflow) (2023.7.22)
Requirement already satisfied: MarkupSafe>=2.1.1 in /Users/sarahsamhithachella/anaconda3/lib/python3.11/site-packages (from wer
kzeug>=1.0.1->tensorboard<2.16,>=2.15->tensorflow-macos==2.15.0->tensorflow) (2.1.1)
Requirement already satisfied: pyasn1<0.5.0,>=0.4.6 in /Users/sarahsamhithachella/anaconda3/lib/python3.11/site-packages (from
pyasn1-modules>=0.2.1->google-auth<3,>=1.6.3->tensorboard<2.16,>=2.15->tensorflow-macos==2.15.0->tensorflow) (0.4.8)
Collecting oauthlib>=3.0.0 (from requests-oauthlib>=0.7.0->google-auth-oauthlib<2,>=0.5->tensorboard<2.16,>=2.15->tensorflow-ma
cos==2.15.0->tensorflow)
  Downloading oauthlib-3.2.2-py3-none-any.whl (151 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 151.7/151.7 kB 10.4 MB/s eta 0:00:00
Downloading tensorflow-2.15.0-cp311-cp311-macosx_12_0_arm64.whl (2.1 kB)
Downloading tensorflow_macos-2.15.0-cp311-cp311-macosx_12_0_arm64.whl (208.8 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 208.8/208.8 MB 2.9 MB/s eta 0:00:0000:0100:02
Downloading absl_py-2.0.0-py3-none-any.whl (130 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 130.2/130.2 kB 3.6 MB/s eta 0:00:00
Downloading flatbuffers-23.5.26-py2.py3-none-any.whl (26 kB)
Downloading grpcio-1.59.3-cp311-cp311-macosx_10_10_universal2.whl (9.6 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 9.6/9.6 MB 3.3 MB/s eta 0:00:0000:0100:01
Downloading keras-2.15.0-py3-none-any.whl (1.7 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.7/1.7 MB 5.0 MB/s eta 0:00:0000:0100:01
Downloading libclang-16.0.6-py2.py3-none-macosx_11_0_arm64.whl (20.6 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 20.6/20.6 MB 3.7 MB/s eta 0:00:0000:0100:01
Downloading ml_dtypes-0.2.0-cp311-cp311-macosx_10_9_universal2.whl (1.2 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.2/1.2 MB 4.6 MB/s eta 0:00:0000:0100:01
```

```
Downloading tensorboard-2.15.1-py3-none-any.whl (5.5 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 5.5/5.5 MB 5.1 MB/s eta 0:00:0000:0100:01
Downloading protobuf-4.23.4-cp37-abi3-macosx_10_9_universal2.whl (400 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 400.3/400.3 kB 5.8 MB/s eta 0:00:00a 0:00:01
Downloading tensorflow_estimator-2.15.0-py2.py3-none-any.whl (441 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 442.0/442.0 kB 5.9 MB/s eta 0:00:00a 0:00:01
Downloading tensorflow_io_gcs_filesystem-0.34.0-cp311-cp311-macosx_12_0_arm64.whl (1.9 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.9/1.9 MB 4.3 MB/s eta 0:00:0000:0100:01
Downloading google_auth-2.23.4-py2.py3-none-any.whl (183 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 183.3/183.3 kB 2.9 MB/s eta 0:00:00a 0:00:01
Downloading google_auth_oauthlib-1.1.0-py2.py3-none-any.whl (19 kB)
Downloading tensorboard_data_server-0.7.2-py3-none-any.whl (2.4 kB)
Downloading cachetools-5.3.2-py3-none-any.whl (9.3 kB)
Installing collected packages: libclang, flatbuffers, termcolor, tensorflow-io-gcs-filesystem, tensorflow-estimator, tensorboar
d-data-server, rsa, protobuf, opt-einsum, oauthlib, ml-dtypes, keras, grpcio, google-pasta, gast, cachetools, astunparse, absl-
py, requests-oauthlib, google-auth, google-auth-oauthlib, tensorboard, tensorflow-macos, tensorflow
Successfully installed absl-py-2.0.0 astunparse-1.6.3 cachetools-5.3.2 flatbuffers-23.5.26 gast-0.5.4 google-auth-2.23.4 google
-auth-oauthlib-1.1.0 google-pasta-0.2.0 grpcio-1.59.3 keras-2.15.0 libclang-16.0.6 ml-dtypes-0.2.0 oauthlib-3.2.2 opt-einsum-3.
3.0 protobuf-4.23.4 requests-oauthlib-1.3.1 rsa-4.9 tensorboard-2.15.1 tensorboard-data-server-0.7.2 tensorflow-2.15.0 tensorfl
ow-estimator-2.15.0 tensorflow-io-gcs-filesystem-0.34.0 tensorflow-macos-2.15.0 termcolor-2.3.0
Note: you may need to restart the kernel to use updated packages.
```

[Image Classification using CNN] Construct a 4-class classification model using a convolutional neural network with the following simple architecture (2 point)]

i 1 Convolutional Layer with 8 3×3 filters.

ii 1 max pooling with 2×2 pool size

iii Flatten the Tensor

iv 1 hidden layer with 16 nodes for fully connected neural network

v Output layer has 4 nodes (since 4 classes) using 'softmax' activation function.

(Use 'Relu' for all layers except the output layer.) for 20 epochs using 'adam' optimizer and 'categorical cross entropy' loss function. If your machine is too slow, you can reduce to 5 epochs. You can perform more epochs (> 20) if you want to. For validation split, you will use 20%. For batch size, you can pick a size that will not slow down the training process on your machine.

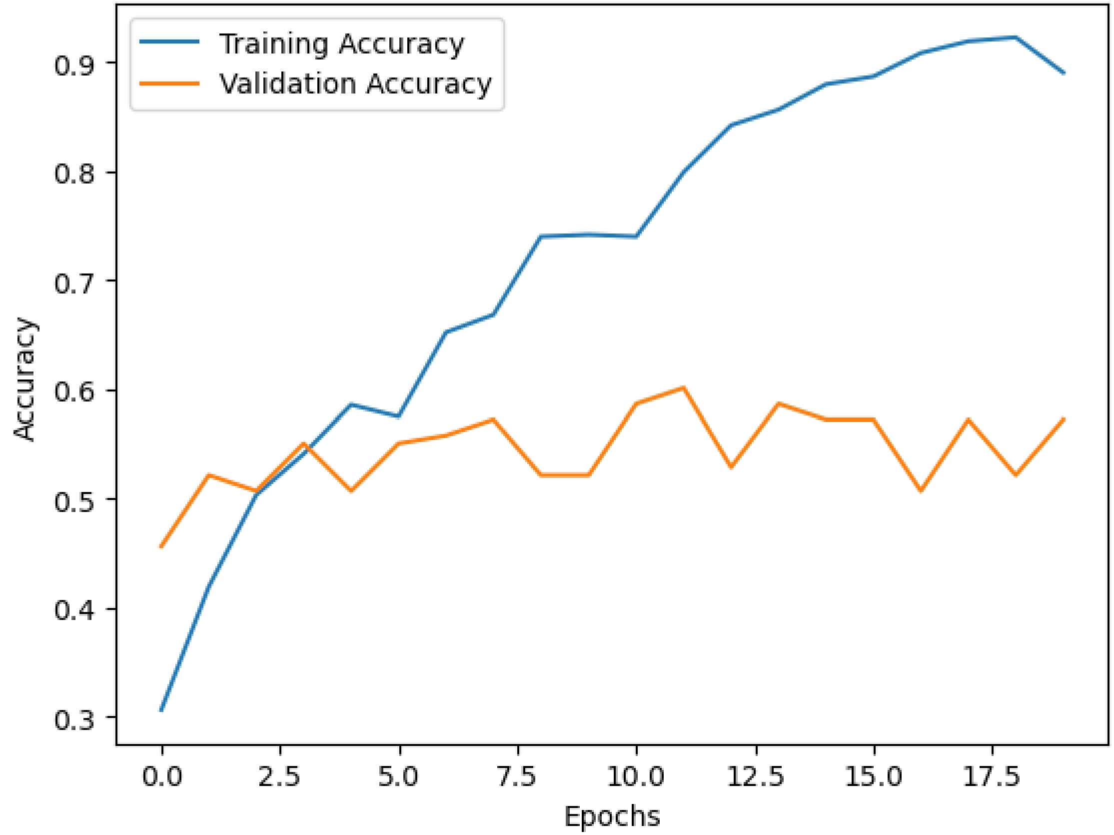
Plot a graph to show the learning curves (i.e., x-axis: number of epochs; y-axis: training andvalidation accuracy - 2 curves) (1 points)

In [9]:
```
import tensorflow as tf
from tensorflow.keras import layers, models
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.optimizers.legacy import Adam as LegacyAdam  # Import the legacy optimizer
from tensorflow.keras.losses import CategoricalCrossentropy
from tensorflow.keras.preprocessing.image import ImageDataGenerator

model = models.Sequential()

# Convolutional Layer with 8 3 × 3 filters
model.add(layers.Conv2D(8, (3, 3), activation='relu', input_shape=(100, 100, 3)))

# MaxPooling Layer with 2 × 2 pool size
model.add(layers.MaxPooling2D((2, 2)))

# Flatten the Tensor
model.add(layers.Flatten())

# Hidden layer with 16 nodes for a fully connected neural network
model.add(layers.Dense(16, activation='relu'))

# Output layer has 4 nodes (since 4 classes) using ‘softmax’ activation function
model.add(layers.Dense(4, activation='softmax'))

model.compile(optimizer=LegacyAdam(), loss=CategoricalCrossentropy(), metrics=['accuracy'])
model.summary()

datagen = ImageDataGenerator(rescale=1./255, validation_split=0.2)

dataset_path = '/Users/sarahsamhithachella/Downloads/SarahNew/Images'

train_generator = datagen.flow_from_directory(
    dataset_path,
    target_size=(100, 100),
    batch_size=32,
    class_mode='categorical',
    subset='training'
)

validation_generator = datagen.flow_from_directory(
    dataset_path,
    target_size=(100, 100),
    batch_size=32,
    class_mode='categorical',
```

```
    subset='validation'
)

history = model.fit(
    train_generator,
    epochs=20,
    validation_data=validation_generator
)

import matplotlib.pyplot as plt

plt.plot(history.history['accuracy'], label='Training Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()
plt.show()
```

```
Model: "sequential_5"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 conv2d_5 (Conv2D)           (None, 98, 98, 8)         224       
                                                                 
 max_pooling2d_5 (MaxPoolin  (None, 49, 49, 8)         0         
 g2D)                                                            
                                                                 
 flatten_5 (Flatten)         (None, 19208)             0         
                                                                 
 dense_10 (Dense)            (None, 16)                307344    
                                                                 
 dense_11 (Dense)            (None, 4)                 68        
                                                                 
=================================================================
Total params: 307636 (1.17 MB)
Trainable params: 307636 (1.17 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
Found 558 images belonging to 4 classes.
Found 138 images belonging to 4 classes.
Epoch 1/20
18/18 [==============================] - 1s 26ms/step - loss: 1.4396 - accuracy: 0.3065 - val_loss: 1.2241 - val_accuracy: 0.45
65
Epoch 2/20
18/18 [==============================] - 0s 16ms/step - loss: 1.2190 - accuracy: 0.4194 - val_loss: 1.1595 - val_accuracy: 0.52
17
Epoch 3/20
18/18 [==============================] - 0s 17ms/step - loss: 1.1541 - accuracy: 0.5036 - val_loss: 1.1416 - val_accuracy: 0.50
72
Epoch 4/20
18/18 [==============================] - 0s 17ms/step - loss: 1.0974 - accuracy: 0.5412 - val_loss: 1.1257 - val_accuracy: 0.55
07
Epoch 5/20
18/18 [==============================] - 0s 16ms/step - loss: 1.0423 - accuracy: 0.5860 - val_loss: 1.1333 - val_accuracy: 0.50
72
Epoch 6/20
18/18 [==============================] - 0s 16ms/step - loss: 1.0419 - accuracy: 0.5753 - val_loss: 1.1038 - val_accuracy: 0.55
07
Epoch 7/20
18/18 [==============================] - 0s 16ms/step - loss: 0.9614 - accuracy: 0.6523 - val_loss: 1.1156 - val_accuracy: 0.55
80
Epoch 8/20
18/18 [==============================] - 0s 16ms/step - loss: 0.9240 - accuracy: 0.6685 - val_loss: 1.0918 - val_accuracy: 0.57
25
Epoch 9/20
18/18 [==============================] - 0s 16ms/step - loss: 0.8566 - accuracy: 0.7401 - val_loss: 1.1313 - val_accuracy: 0.52
```

```
17
Epoch 10/20
18/18 [==============================] - 0s 16ms/step - loss: 0.8387 - accuracy: 0.7419 - val_loss: 1.1345 - val_accuracy: 0.52
17
Epoch 11/20
18/18 [==============================] - 0s 16ms/step - loss: 0.8274 - accuracy: 0.7401 - val_loss: 1.0905 - val_accuracy: 0.58
70
Epoch 12/20
18/18 [==============================] - 0s 16ms/step - loss: 0.7654 - accuracy: 0.7993 - val_loss: 1.0889 - val_accuracy: 0.60
14
Epoch 13/20
18/18 [==============================] - 0s 18ms/step - loss: 0.6982 - accuracy: 0.8423 - val_loss: 1.1765 - val_accuracy: 0.52
90
Epoch 14/20
18/18 [==============================] - 0s 16ms/step - loss: 0.6594 - accuracy: 0.8566 - val_loss: 1.1244 - val_accuracy: 0.58
70
Epoch 15/20
18/18 [==============================] - 0s 16ms/step - loss: 0.6129 - accuracy: 0.8799 - val_loss: 1.1821 - val_accuracy: 0.57
25
Epoch 16/20
18/18 [==============================] - 0s 16ms/step - loss: 0.5912 - accuracy: 0.8871 - val_loss: 1.1876 - val_accuracy: 0.57
25
Epoch 17/20
18/18 [==============================] - 0s 16ms/step - loss: 0.5489 - accuracy: 0.9086 - val_loss: 1.2643 - val_accuracy: 0.50
72
Epoch 18/20
18/18 [==============================] - 0s 16ms/step - loss: 0.5159 - accuracy: 0.9194 - val_loss: 1.1857 - val_accuracy: 0.57
25
Epoch 19/20
18/18 [==============================] - 0s 17ms/step - loss: 0.4900 - accuracy: 0.9229 - val_loss: 1.3795 - val_accuracy: 0.52
17
Epoch 20/20
18/18 [==============================] - 0s 16ms/step - loss: 0.5184 - accuracy: 0.8907 - val_loss: 1.2610 - val_accuracy: 0.57
25
```



Perform ONE of the following experiment below ((a), (b) or (c)) based on the last digit of your Rowan Banner ID (1 point): (a) Train the CNN using 2 other filter sizes: 5 × 5 and 7 × 7 for the convolution layer (i) with all other parameters unchanged (b) Train the CNN using 2 other number of filters: 4 and 16 for the convolution layer (i) with all other parameters unchanged (c) Train the CNN using 2 other number of nodes in the hidden layer (iv): 8 and 32 with all other parameters unchanged If the last digit is {0, 1, 2, 3}, do (a). If the last digit is {4, 5, 6}, do (b). If the last digit is {7, 8, 9}, do (c). State your Rowan Banner ID in your submission so that we know which experiment you are doing.

Plot the learning curves (i.e., x-axis: number of epochs; y-axis: training and validation accuracy -2 curves) for the classification models using the above 2 different parameter values (1 points)

BANNER ID: 916451763 Since the last digit of my banner iD is 3. Do (a)

(a) Train the CNN using 2 other filter sizes:

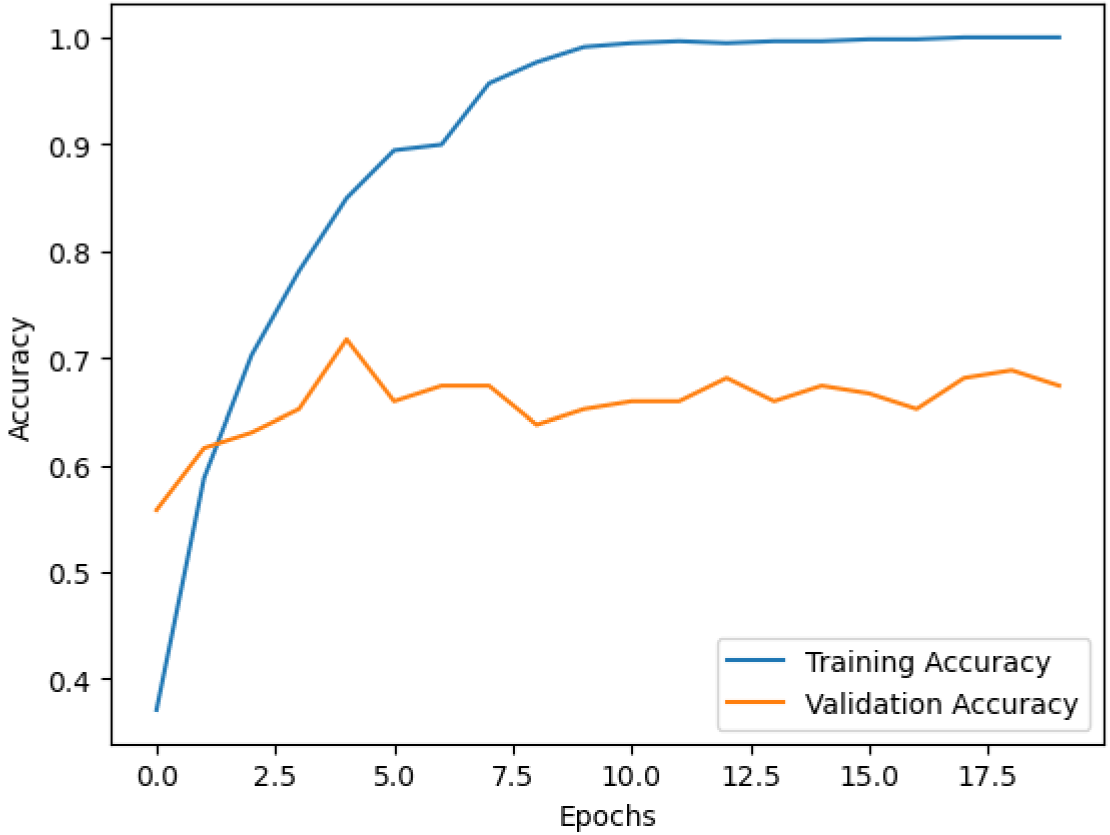
5 × 5 for the convolution layer (i) with all other parameters unchanged

In [10]:
```
import tensorflow as tf
from tensorflow.keras import layers, models
from tensorflow.keras.optimizers.legacy import Adam as LegacyAdam
from tensorflow.keras.losses import CategoricalCrossentropy
from tensorflow.keras.preprocessing.image import ImageDataGenerator
import matplotlib.pyplot as plt

model = models.Sequential()

# Convolutional Layer with 2 other filter sizes: 5 × 5 and 7 × 7
model.add(layers.Conv2D(8, (5, 5), activation='relu', input_shape=(100, 100, 3)))
# Alternatively, you can uncomment the line below to use 7 × 7 filters

# MaxPooling Layer with 2 × 2 pool size
model.add(layers.MaxPooling2D((2, 2)))

# Flatten the Tensor
model.add(layers.Flatten())

# Hidden layer with 16 nodes for a fully connected neural network
model.add(layers.Dense(16, activation='relu'))

# Output layer has 4 nodes (since 4 classes) using ‘softmax’ activation function
model.add(layers.Dense(4, activation='softmax'))

model.compile(optimizer=LegacyAdam(), loss=CategoricalCrossentropy(), metrics=['accuracy'])
model.summary()

datagen = ImageDataGenerator(rescale=1./255, validation_split=0.2)

dataset_path = '/Users/sarahsamhithachella/Downloads/SarahNew/Images'

train_generator = datagen.flow_from_directory(
    dataset_path,
    target_size=(100, 100),
    batch_size=32,
    class_mode='categorical',
    subset='training'
)

validation_generator = datagen.flow_from_directory(
    dataset_path,
    target_size=(100, 100),
    batch_size=32,
    class_mode='categorical',
    subset='validation'
```

```python
)

history = model.fit(
    train_generator,
    epochs=20,
    validation_data=validation_generator
)

import matplotlib.pyplot as plt

# Plot the learning curves
plt.plot(history.history['accuracy'], label='Training Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()
plt.show()
```

```
Model: "sequential_6"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 conv2d_6 (Conv2D)           (None, 96, 96, 8)         608       
                                                                 
 max_pooling2d_6 (MaxPoolin  (None, 48, 48, 8)         0         
 g2D)                                                            
                                                                 
 flatten_6 (Flatten)         (None, 18432)             0         
                                                                 
 dense_12 (Dense)            (None, 16)                294928    
                                                                 
 dense_13 (Dense)            (None, 4)                 68        
                                                                 
=================================================================
Total params: 295604 (1.13 MB)
Trainable params: 295604 (1.13 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
Found 558 images belonging to 4 classes.
Found 138 images belonging to 4 classes.
Epoch 1/20
18/18 [==============================] - 1s 31ms/step - loss: 1.3017 - accuracy: 0.3710 - val_loss: 1.0819 - val_accuracy: 0.55
80
Epoch 2/20
18/18 [==============================] - 1s 29ms/step - loss: 0.9764 - accuracy: 0.5878 - val_loss: 0.8957 - val_accuracy: 0.61
59
Epoch 3/20
18/18 [==============================] - 1s 29ms/step - loss: 0.7561 - accuracy: 0.7025 - val_loss: 0.8469 - val_accuracy: 0.63
04
Epoch 4/20
18/18 [==============================] - 1s 29ms/step - loss: 0.6190 - accuracy: 0.7814 - val_loss: 0.7844 - val_accuracy: 0.65
22
Epoch 5/20
18/18 [==============================] - 1s 30ms/step - loss: 0.4726 - accuracy: 0.8495 - val_loss: 0.7389 - val_accuracy: 0.71
74
Epoch 6/20
18/18 [==============================] - 1s 29ms/step - loss: 0.3857 - accuracy: 0.8943 - val_loss: 0.7474 - val_accuracy: 0.65
94
Epoch 7/20
18/18 [==============================] - 1s 28ms/step - loss: 0.3426 - accuracy: 0.8996 - val_loss: 0.7668 - val_accuracy: 0.67
39
Epoch 8/20
18/18 [==============================] - 1s 29ms/step - loss: 0.2542 - accuracy: 0.9570 - val_loss: 0.8622 - val_accuracy: 0.67
39
Epoch 9/20
18/18 [==============================] - 1s 29ms/step - loss: 0.1880 - accuracy: 0.9767 - val_loss: 0.8033 - val_accuracy: 0.63
```

```
77
Epoch 10/20
18/18 [==============================] - 1s 29ms/step - loss: 0.1292 - accuracy: 0.9910 - val_loss: 0.8020 - val_accuracy: 0.65
22
Epoch 11/20
18/18 [==============================] - 1s 29ms/step - loss: 0.1050 - accuracy: 0.9946 - val_loss: 0.9037 - val_accuracy: 0.65
94
Epoch 12/20
18/18 [==============================] - 1s 28ms/step - loss: 0.0816 - accuracy: 0.9964 - val_loss: 0.8750 - val_accuracy: 0.65
94
Epoch 13/20
18/18 [==============================] - 1s 30ms/step - loss: 0.0639 - accuracy: 0.9946 - val_loss: 0.9106 - val_accuracy: 0.68
12
Epoch 14/20
18/18 [==============================] - 1s 29ms/step - loss: 0.0585 - accuracy: 0.9964 - val_loss: 0.9149 - val_accuracy: 0.65
94
Epoch 15/20
18/18 [==============================] - 1s 29ms/step - loss: 0.0460 - accuracy: 0.9964 - val_loss: 0.9544 - val_accuracy: 0.67
39
Epoch 16/20
18/18 [==============================] - 1s 29ms/step - loss: 0.0366 - accuracy: 0.9982 - val_loss: 0.9511 - val_accuracy: 0.66
67
Epoch 17/20
18/18 [==============================] - 1s 29ms/step - loss: 0.0299 - accuracy: 0.9982 - val_loss: 1.0472 - val_accuracy: 0.65
22
Epoch 18/20
18/18 [==============================] - 1s 29ms/step - loss: 0.0252 - accuracy: 1.0000 - val_loss: 1.0587 - val_accuracy: 0.68
12
Epoch 19/20
18/18 [==============================] - 1s 29ms/step - loss: 0.0215 - accuracy: 1.0000 - val_loss: 1.0160 - val_accuracy: 0.68
84
Epoch 20/20
18/18 [==============================] - 1s 30ms/step - loss: 0.0187 - accuracy: 1.0000 - val_loss: 1.0307 - val_accuracy: 0.67
39
```



(a)Train the CNN using 2 other filter sizes:

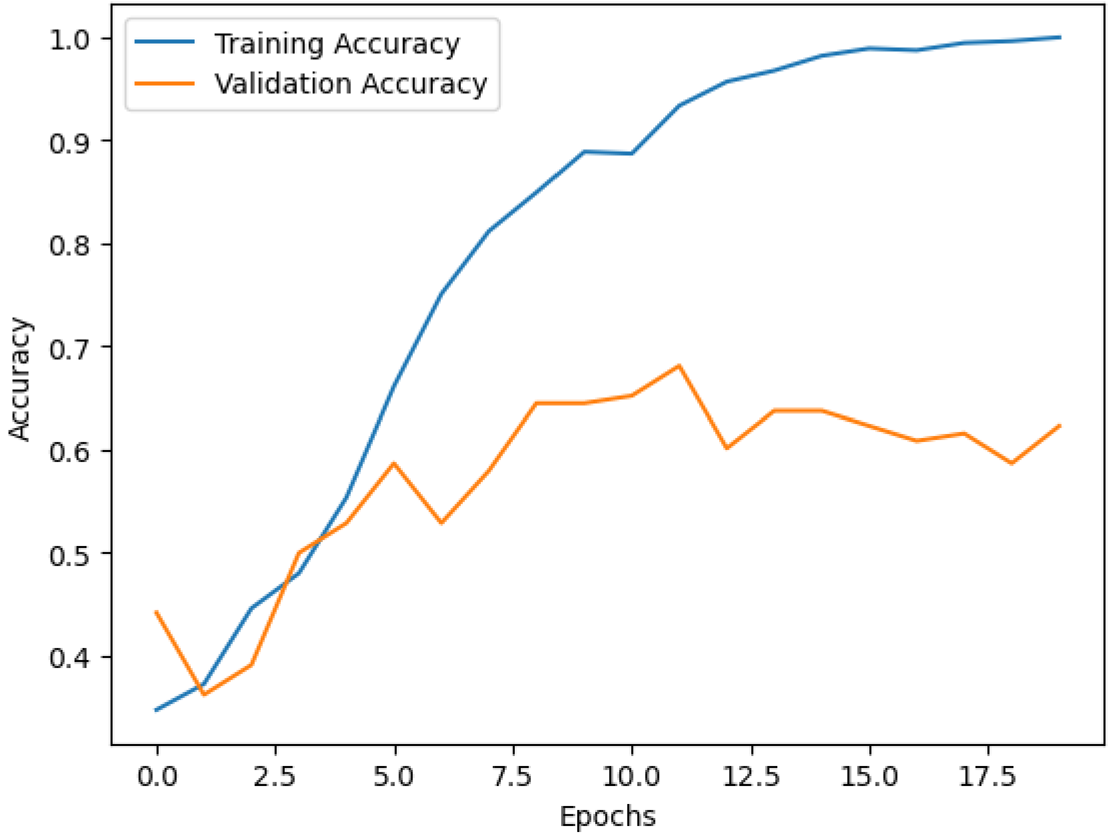
7 × 7 for the convolution layer (i) with all other parameters unchanged

In [11]:
```
import tensorflow as tf
from tensorflow.keras import layers, models
from tensorflow.keras.optimizers.legacy import Adam as LegacyAdam
from tensorflow.keras.losses import CategoricalCrossentropy
from tensorflow.keras.preprocessing.image import ImageDataGenerator
import matplotlib.pyplot as plt

model = models.Sequential()

model.add(layers.Conv2D(8, (7, 7), activation='relu', input_shape=(100, 100, 3)))

# MaxPooling Layer with 2 × 2 pool size
model.add(layers.MaxPooling2D((2, 2)))

# Flatten the Tensor
```

```python
model.add(layers.Flatten())

# Hidden layer with 16 nodes for a fully connected neural network
model.add(layers.Dense(16, activation='relu'))

# Output layer has 4 nodes (since 4 classes) using ‘softmax’ activation function
model.add(layers.Dense(4, activation='softmax'))

model.compile(optimizer=LegacyAdam(), loss=CategoricalCrossentropy(), metrics=['accuracy'])
model.summary()

datagen = ImageDataGenerator(rescale=1./255, validation_split=0.2)

dataset_path = '/Users/sarahsamhithachella/Downloads/SarahNew/Images'

train_generator = datagen.flow_from_directory(
    dataset_path,
    target_size=(100, 100),
    batch_size=32,
    class_mode='categorical',
    subset='training'
)

validation_generator = datagen.flow_from_directory(
    dataset_path,
    target_size=(100, 100),
    batch_size=32,
    class_mode='categorical',
    subset='validation'
)

history = model.fit(
    train_generator,
    epochs=20,
    validation_data=validation_generator
)

import matplotlib.pyplot as plt

# Plot the learning curves
plt.plot(history.history['accuracy'], label='Training Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()
plt.show()
```

```
Model: "sequential_7"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_7 (Conv2D)           (None, 94, 94, 8)         1184

 max_pooling2d_7 (MaxPoolin  (None, 47, 47, 8)         0
 g2D)

 flatten_7 (Flatten)         (None, 17672)             0

 dense_14 (Dense)            (None, 16)                282768

 dense_15 (Dense)            (None, 4)                 68

=================================================================
Total params: 284020 (1.08 MB)
Trainable params: 284020 (1.08 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
Found 558 images belonging to 4 classes.
Found 138 images belonging to 4 classes.
Epoch 1/20
18/18 [==============================] - 1s 47ms/step - loss: 1.4177 - accuracy: 0.3477 - val_loss: 1.3098 - val_accuracy: 0.44
20
Epoch 2/20
18/18 [==============================] - 1s 45ms/step - loss: 1.2172 - accuracy: 0.3728 - val_loss: 1.1820 - val_accuracy: 0.36
23
Epoch 3/20
18/18 [==============================] - 1s 43ms/step - loss: 1.1226 - accuracy: 0.4462 - val_loss: 1.1599 - val_accuracy: 0.39
13
Epoch 4/20
18/18 [==============================] - 1s 43ms/step - loss: 1.0635 - accuracy: 0.4803 - val_loss: 1.1818 - val_accuracy: 0.50
00
Epoch 5/20
18/18 [==============================] - 1s 43ms/step - loss: 0.9948 - accuracy: 0.5538 - val_loss: 1.0836 - val_accuracy: 0.52
90
Epoch 6/20
18/18 [==============================] - 1s 44ms/step - loss: 0.8354 - accuracy: 0.6613 - val_loss: 0.9492 - val_accuracy: 0.58
70
Epoch 7/20
18/18 [==============================] - 1s 43ms/step - loss: 0.6750 - accuracy: 0.7509 - val_loss: 1.0837 - val_accuracy: 0.52
90
Epoch 8/20
18/18 [==============================] - 1s 43ms/step - loss: 0.5814 - accuracy: 0.8118 - val_loss: 1.0857 - val_accuracy: 0.57
97
Epoch 9/20
18/18 [==============================] - 1s 43ms/step - loss: 0.5004 - accuracy: 0.8495 - val_loss: 0.9227 - val_accuracy: 0.64
```

```
49
Epoch 10/20
18/18 [==============================] - 1s 44ms/step - loss: 0.4080 - accuracy: 0.8889 - val_loss: 0.8945 - val_accuracy: 0.64
49
Epoch 11/20
18/18 [==============================] - 1s 44ms/step - loss: 0.3656 - accuracy: 0.8871 - val_loss: 0.9764 - val_accuracy: 0.65
22
Epoch 12/20
18/18 [==============================] - 1s 43ms/step - loss: 0.2946 - accuracy: 0.9337 - val_loss: 0.9842 - val_accuracy: 0.68
12
Epoch 13/20
18/18 [==============================] - 1s 44ms/step - loss: 0.2254 - accuracy: 0.9570 - val_loss: 1.1202 - val_accuracy: 0.60
14
Epoch 14/20
18/18 [==============================] - 1s 43ms/step - loss: 0.1818 - accuracy: 0.9677 - val_loss: 1.0346 - val_accuracy: 0.63
77
Epoch 15/20
18/18 [==============================] - 1s 43ms/step - loss: 0.1321 - accuracy: 0.9821 - val_loss: 1.1495 - val_accuracy: 0.63
77
Epoch 16/20
18/18 [==============================] - 1s 45ms/step - loss: 0.1059 - accuracy: 0.9892 - val_loss: 1.1922 - val_accuracy: 0.62
32
Epoch 17/20
18/18 [==============================] - 1s 43ms/step - loss: 0.0876 - accuracy: 0.9875 - val_loss: 1.2454 - val_accuracy: 0.60
87
Epoch 18/20
18/18 [==============================] - 1s 43ms/step - loss: 0.0785 - accuracy: 0.9946 - val_loss: 1.2257 - val_accuracy: 0.61
59
Epoch 19/20
18/18 [==============================] - 1s 43ms/step - loss: 0.0640 - accuracy: 0.9964 - val_loss: 1.4161 - val_accuracy: 0.58
70
Epoch 20/20
18/18 [==============================] - 1s 43ms/step - loss: 0.0422 - accuracy: 1.0000 - val_loss: 1.3145 - val_accuracy: 0.62
32
```



Describe and discuss what you observe by comparing the performance of the first model and the other two models you constructed in (a), (b) or (c) (depending on which one you did). Are theremodel overfit or underfit or just right? (1 point)

Model: "sequential_5"

Architecture: Convolutional Neural Network with 1 convolutional layer, max pooling, and 2 dense layers.

Total params: 307,636

Training accuracy: ~89.1%

Validation accuracy: ~57.3%

Loss: After about 12 epochs, the validation loss slightly increases while the training loss gradually decreases. So, there might be overfitting.

Model: "sequential_6"

Architecture: CNN with 1 convolutional layer, max pooling, and 2 dense layers.

Total params: 295,604

Training accuracy: ~100%

Validation accuracy: ~67.4%

Loss: Both training and validation losses decrease consistently. This indicates a good fit and no signs of overfitting.

Model: "sequential_7"

Architecture: CNN with 1 convolutional layer, max pooling, and 2 dense layers.

Total params: 284,020

Training accuracy: ~100%

Validation accuracy: ~62.3%

Loss: Similar to Model 6, both training and validation losses decrease consistently. This also indicates a good fit and no signs of overfitting.

Is the model overfit or underfit or just right? :

(Model 5): As the training accuracy is significantly higher than the validation accuracy, and the validation loss increases after a certain point, model 5 shows signs of overfitting.

Model 5 has the highest number of parameters, which might contribute to overfitting.

Model 5 lags behind in terms of validation accuracy.

(Models 6 and 7): These models do not exhibit clear signs of overfitting. Both training and validation accuracies are high, and losses continue to decrease.

Models 6 and 7 have slightly fewer parameters but still achieve high accuracy, indicating that they might be more efficient in this context.

Validation Accuracy:

Model 6 has the highest validation accuracy, followed closely by Model 7.

Model 5 lags behind in terms of validation accuracy.

References:

Java point: Association Rule Learning

Association Rules With Python

CNN Image Classification

Heatmap using Seaborn

Simple MNIST Convnet

Association Rules Guide

```
In [ ]:
```