In [2]:
```
#Dataset : Sea Animals Image Dataset
#https://www.kaggle.com/datasets/vencerlanz09/sea-animals-image-dataste?select=Jelly+Fish
# Dolphin - Jelly Fish - Seahorse - Sharks - Turtle_Tortoise
```

In [3]:
```
#Split

import os, shutil, random

dataset_path = "/Users/fm/Desktop/Project 1/archive"
output_base = "sea_animals_5classes_split"

train_ratio, val_ratio, test_ratio = 0.7, 0.15, 0.15

os.makedirs(output_base, exist_ok=True)
for folder in ["train", "val", "test"]:
    os.makedirs(os.path.join(output_base, folder), exist_ok=True)

# Only folders (your 5 classes)
classes = [c for c in os.listdir(dataset_path)
           if os.path.isdir(os.path.join(dataset_path, c))]

print("Classes found:", classes)

for cls in classes:
    cls_path = os.path.join(dataset_path, cls)
    images = [img for img in os.listdir(cls_path)
              if os.path.isfile(os.path.join(cls_path, img))]

    random.shuffle(images)

    train_end = int(len(images) * train_ratio)
    val_end = train_end + int(len(images) * val_ratio)

    train_imgs = images[:train_end]
    val_imgs = images[train_end:val_end]
    test_imgs = images[val_end:]

    for split, split_imgs in zip(["train", "val", "test"],
                                 [train_imgs, val_imgs, test_imgs]):
        split_dir = os.path.join(output_base, split, cls)
        os.makedirs(split_dir, exist_ok=True)

        for img in split_imgs:
            shutil.copy(os.path.join(cls_path, img),
                        os.path.join(split_dir, img))

print("\n DONE: Data split completed!")
```

```
Classes found: ['Seahorse', 'Turtle_Tortoise', 'Dolphin', 'Jelly Fish', 'Sharks']

 DONE: Data split completed!
```

In [6]:
```
#Augmentation

from tensorflow.keras.preprocessing.image import ImageDataGenerator

train_gen = ImageDataGenerator(
    rescale=1./255,
    rotation_range=20,
    width_shift_range=0.2,
    height_shift_range=0.2,
    zoom_range=0.2,
    brightness_range=[0.8, 1.2],
    horizontal_flip=True
)

val_test_gen = ImageDataGenerator(rescale=1./255)

train_data = train_gen.flow_from_directory(
    "sea_animals_5classes_split/train",
    target_size=(224, 224),
    batch_size=32,
    class_mode="categorical"

)

val_data = val_test_gen.flow_from_directory(
    "sea_animals_5classes_split/val",
    target_size=(224, 224),
    batch_size=32,
class_mode="categorical"
)

test_data = val_test_gen.flow_from_directory(
    "sea_animals_5classes_split/test",
    target_size=(224, 224),
    batch_size=32,
    class_mode="categorical",
    shuffle=False
)
```

```
Found 3217 images belonging to 5 classes.
Found 687 images belonging to 5 classes.
Found 694 images belonging to 5 classes.
```

In [10]:
```
#Training and testing models - MobileNetV2

import tensorflow as tf
from tensorflow.keras.applications import MobileNetV2
from tensorflow.keras import layers, models

num_classes = len(train_data.class_indices)

# Load MobileNetV2 base model
base = MobileNetV2(
    include_top=False,
    weights="imagenet",
    input_shape=(224, 224, 3)
)

#  Freeze base model layers
base.trainable = False
```

```python
#  Build full model
inputs = layers.Input(shape=(224, 224, 3))
x = base(inputs, training=False)
x = layers.GlobalAveragePooling2D()(x)
x = layers.Dropout(0.4)(x)
outputs = layers.Dense(num_classes, activation="softmax")(x)

model = models.Model(inputs, outputs)

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

#  Train
history = model.fit(
    train_data,
    validation_data=val_data,
    epochs=5
)

#  Save model
model.save("mobilenetv2_sea_animals.h5")
```

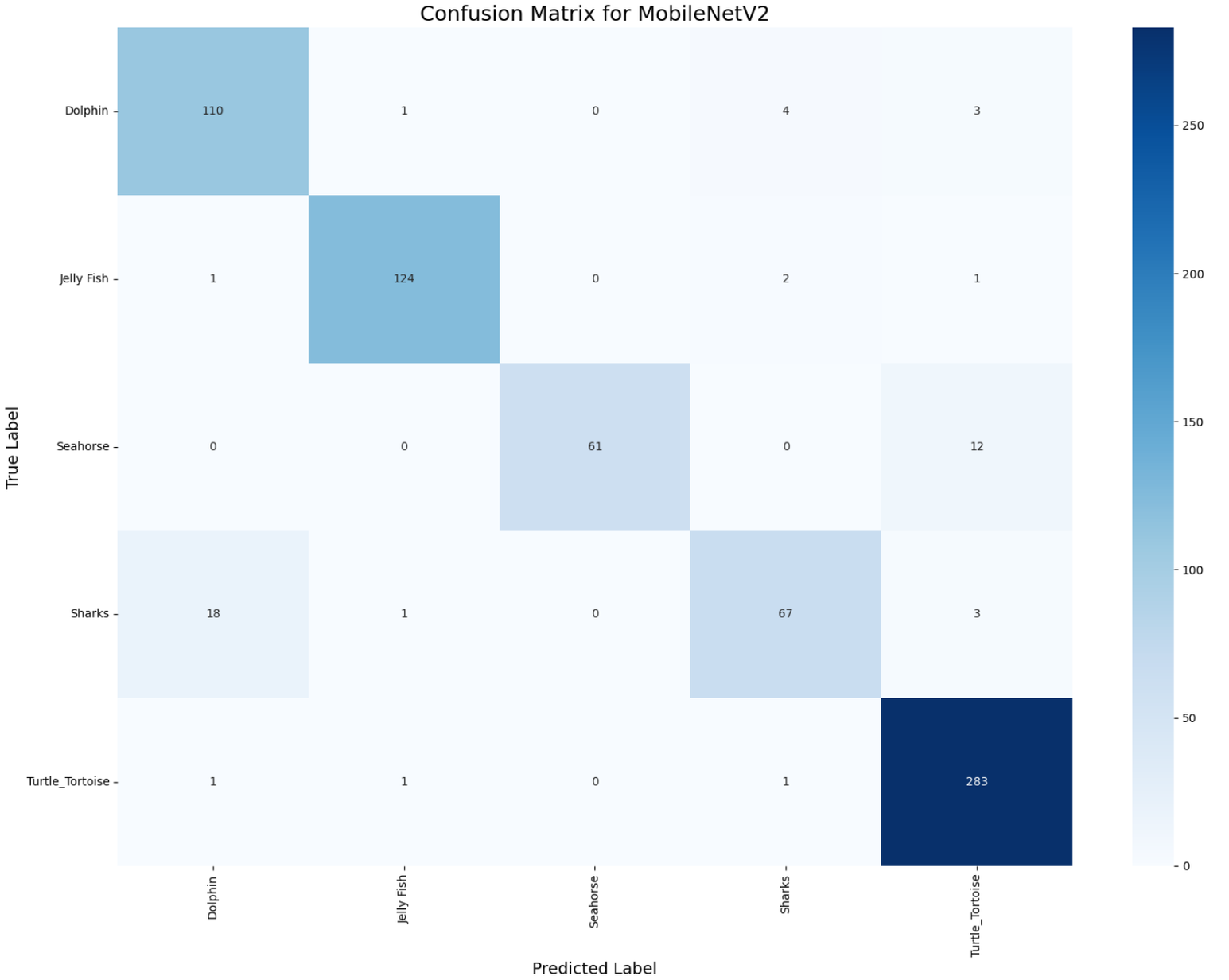
```
Epoch 1/5
101/101 ━━━━━━━━━━━━━━━━━━━━ 87s 819ms/step - accuracy: 0.7153 - loss: 0.7611 - val_accuracy: 0.8850 - val_loss: 0.3352
Epoch 2/5
101/101 ━━━━━━━━━━━━━━━━━━━━ 80s 786ms/step - accuracy: 0.8542 - loss: 0.3974 - val_accuracy: 0.9112 - val_loss: 0.2651
Epoch 3/5
101/101 ━━━━━━━━━━━━━━━━━━━━ 81s 798ms/step - accuracy: 0.8813 - loss: 0.3328 - val_accuracy: 0.9156 - val_loss: 0.2474
Epoch 4/5
101/101 ━━━━━━━━━━━━━━━━━━━━ 81s 800ms/step - accuracy: 0.8934 - loss: 0.2993 - val_accuracy: 0.9068 - val_loss: 0.2548
Epoch 5/5
101/101 ━━━━━━━━━━━━━━━━━━━━ 82s 807ms/step - accuracy: 0.9024 - loss: 0.2795 - val_accuracy: 0.9185 - val_loss: 0.2363
```

```
WARNING:absl:You are saving your model as an HDF5 file via `model.save()` or `keras.saving.save_model(model)`. This file format is considered legacy. We recommend using instead the native Keras format, e.g. `model.save('my_model.keras')` or `keras.saving.save_model(model, 'my_model.keras')`.
```

In [11]:

```python
from sklearn.metrics import (
    classification_report,
    accuracy_score,
    precision_score,
    recall_score,
    f1_score,
    confusion_matrix
)
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns


# Predict probabilities
y_pred = model.predict(test_data)

# Convert probabilities → class indices
y_pred_classes = np.argmax(y_pred, axis=1)

# True labels
y_true = test_data.classes

# Class names
class_labels = list(test_data.class_indices.keys())


accuracy = accuracy_score(y_true, y_pred_classes)
precision = precision_score(y_true, y_pred_classes, average='weighted')
recall = recall_score(y_true, y_pred_classes, average='weighted')
f1 = f1_score(y_true, y_pred_classes, average='weighted')

print("Accuracy:", accuracy)
print("Precision:", precision)
print("Recall:", recall)
print("F1-score:", f1)


print("\nClassification Report:")
print(classification_report(y_true, y_pred_classes, target_names=class_labels))



cm = confusion_matrix(y_true, y_pred_classes)

plt.figure(figsize=(16, 12))
sns.heatmap(cm, annot=True, fmt="d", cmap="Blues",
            xticklabels=class_labels,
            yticklabels=class_labels)

plt.title("Confusion Matrix for MobileNetV2", fontsize=18)
plt.ylabel("True Label", fontsize=14)
plt.xlabel("Predicted Label", fontsize=14)
plt.xticks(rotation=90)
plt.yticks(rotation=0)
plt.tight_layout()
plt.show()
```

```
/opt/anaconda3/lib/python3.12/site-packages/keras/src/trainers/data_adapters/py_dataset_adapter.py:121: UserWarning: Your `PyDataset` class should call `super().__init__(**kwargs)` in its constructor. `**kwargs` can include `workers`, `use_multiprocessing`, `max_queue_size`. Do not pass these arguments to `fit()`, as they will be ignored.
  self._warn_if_super_not_called()
22/22 ━━━━━━━━━━━━━━━━━━━━ 17s 731ms/step
Accuracy: 0.9293948126801153
Precision: 0.931426852156408
Recall: 0.9293948126801153
F1-score: 0.9280855105640881

Classification Report:
                 precision    recall  f1-score   support

        Dolphin       0.85      0.93      0.89       118
     Jelly Fish       0.98      0.97      0.97       128
       Seahorse       1.00      0.84      0.91        73
         Sharks       0.91      0.75      0.82        89
Turtle_Tortoise       0.94      0.99      0.96       286

       accuracy                           0.93       694
      macro avg       0.93      0.90      0.91       694
   weighted avg       0.93      0.93      0.93       694
```



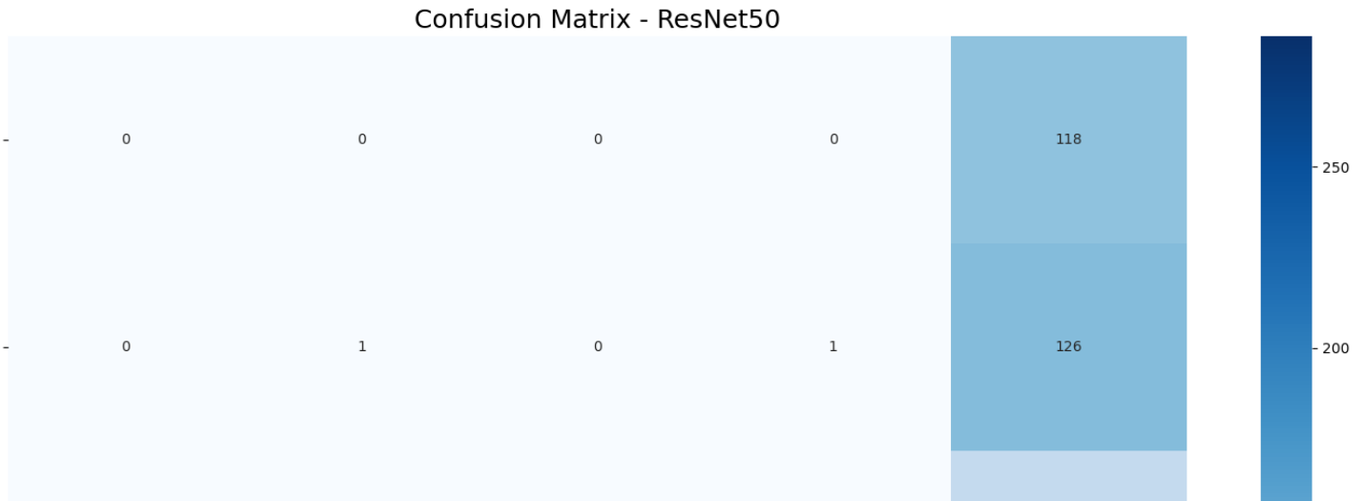
In [12]:

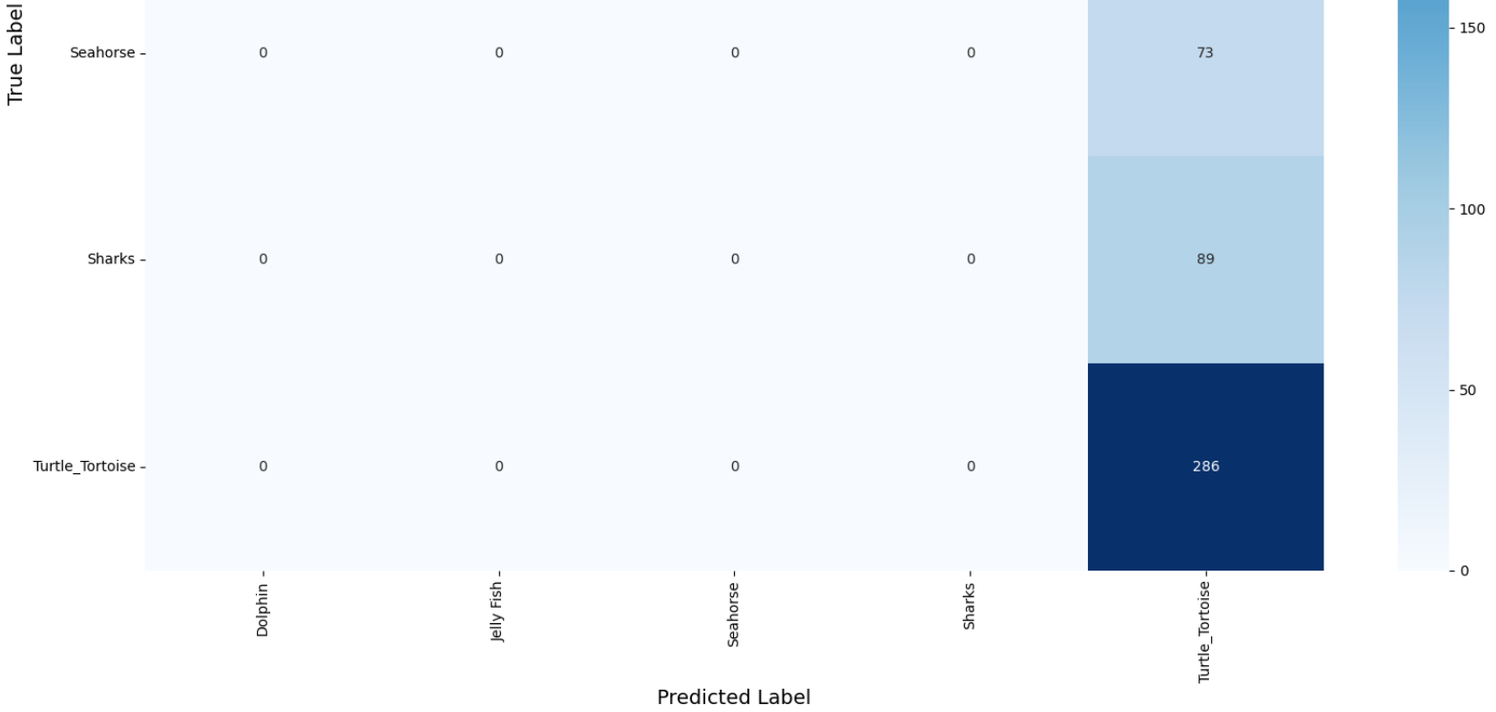
```
#Training and testing models - ResNet50

import tensorflow as tf
from tensorflow.keras.applications import ResNet50
from tensorflow.keras import layers, models
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np

# Number of classes
num_classes = len(train_data.class_indices)

# Load ResNet50
base = ResNet50(
    include_top=False,
    weights="imagenet",
    input_shape=(224, 224, 3)
)

# Freeze base layers
base.trainable = False

# Build the model
inputs = layers.Input(shape=(224, 224, 3))
x = base(inputs, training=False)
x = layers.GlobalAveragePooling2D()(x)
x = layers.Dropout(0.4)(x)
outputs = layers.Dense(num_classes, activation="softmax")(x)

model_resnet = models.Model(inputs, outputs)

# Compile
model_resnet.compile(
    optimizer="adam",
    loss="categorical_crossentropy",
    metrics=["accuracy"]
)

# Train model
history_resnet = model_resnet.fit(
    train_data,
    validation_data=val_data,
    epochs=5
)

# Save
model_resnet.save("resnet50_sea_animals.keras")

#---------------------------------
```

```python
from sklearn.metrics import (
    classification_report,
    accuracy_score,
    precision_score,
    recall_score,
    f1_score,
    confusion_matrix
)


# Predictions
y_pred = model_resnet.predict(test_data)
y_pred_classes = np.argmax(y_pred, axis=1)

# True labels
y_true = test_data.classes

# Class names
class_labels = list(test_data.class_indices.keys())

# Metrics
accuracy = accuracy_score(y_true, y_pred_classes)
precision = precision_score(y_true, y_pred_classes, average='weighted')
recall = recall_score(y_true, y_pred_classes, average='weighted')
f1 = f1_score(y_true, y_pred_classes, average='weighted')

print("Accuracy:", accuracy)
print("Precision:", precision)
print("Recall:", recall)
print("F1-score:", f1)

print("\nClassification Report:")
print(classification_report(y_true, y_pred_classes, target_names=class_labels))

#--------------------------------

cm = confusion_matrix(y_true, y_pred_classes)

plt.figure(figsize=(16, 12))
sns.heatmap(cm, annot=True, fmt="d", cmap="Blues",
            xticklabels=class_labels,
            yticklabels=class_labels)

plt.title("Confusion Matrix - ResNet50", fontsize=18)
plt.ylabel("True Label", fontsize=14)
plt.xlabel("Predicted Label", fontsize=14)
plt.xticks(rotation=90)
plt.yticks(rotation=0)
plt.tight_layout()
plt.show()
```

```
Epoch 1/5
101/101 ━━━━━━━━━━━━━━━━━━━━ 186s 2s/step - accuracy: 0.3817 - loss: 1.5705 - val_accuracy: 0.4148 - val_loss: 1.3996
Epoch 2/5
101/101 ━━━━━━━━━━━━━━━━━━━━ 172s 2s/step - accuracy: 0.4137 - loss: 1.4447 - val_accuracy: 0.4192 - val_loss: 1.3605
Epoch 3/5
101/101 ━━━━━━━━━━━━━━━━━━━━ 172s 2s/step - accuracy: 0.4206 - loss: 1.4040 - val_accuracy: 0.4148 - val_loss: 1.3268
Epoch 4/5
101/101 ━━━━━━━━━━━━━━━━━━━━ 173s 2s/step - accuracy: 0.4349 - loss: 1.3748 - val_accuracy: 0.4178 - val_loss: 1.3547
Epoch 5/5
101/101 ━━━━━━━━━━━━━━━━━━━━ 171s 2s/step - accuracy: 0.4454 - loss: 1.3648 - val_accuracy: 0.4163 - val_loss: 1.3338
22/22 ━━━━━━━━━━━━━━━━━━━━ 29s 1s/step
Accuracy: 0.41354466858789624
Precision: 0.35475837483966616
Recall: 0.41354466858789624
F1-score: 0.24388541752754006

Classification Report:
                 precision    recall  f1-score   support

        Dolphin       0.00      0.00      0.00       118
     Jelly Fish       1.00      0.01      0.02       128
       Seahorse       0.00      0.00      0.00        73
         Sharks       0.00      0.00      0.00        89
Turtle_Tortoise       0.41      1.00      0.58       286

       accuracy                           0.41       694
      macro avg       0.28      0.20      0.12       694
   weighted avg       0.35      0.41      0.24       694
```

```
/opt/anaconda3/lib/python3.12/site-packages/sklearn/metrics/_classification.py:1531: UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 in labels wit
h no predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/opt/anaconda3/lib/python3.12/site-packages/sklearn/metrics/_classification.py:1531: UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 in labels wit
h no predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/opt/anaconda3/lib/python3.12/site-packages/sklearn/metrics/_classification.py:1531: UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 in labels wit
h no predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/opt/anaconda3/lib/python3.12/site-packages/sklearn/metrics/_classification.py:1531: UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 in labels wit
h no predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
```





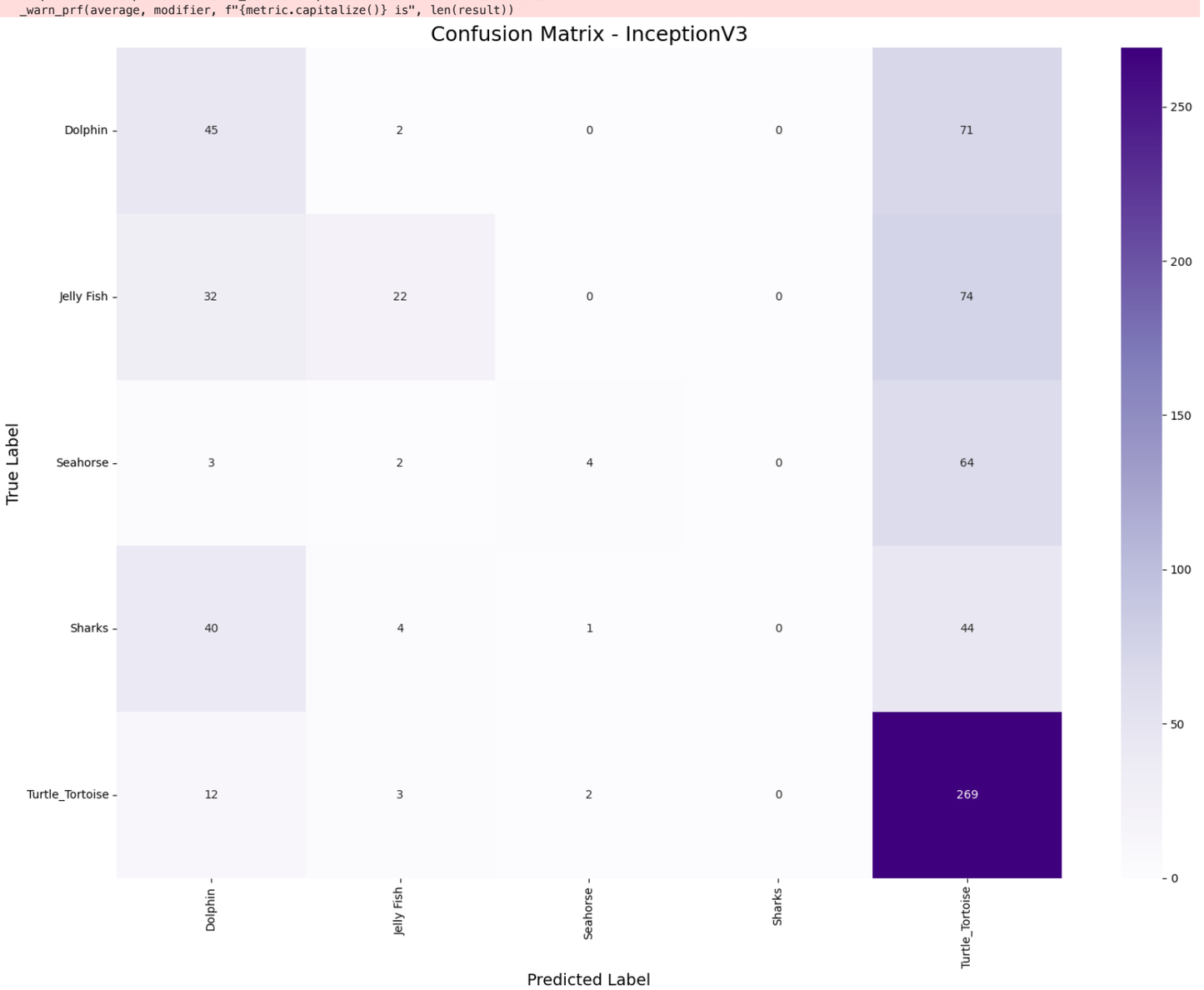
In [14]:

```
#Training and testing models –InceptionV3


import tensorflow as tf
from tensorflow.keras.applications import InceptionV3
from tensorflow.keras.applications.inception_v3 import preprocess_input
from tensorflow.keras import layers, models
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

num_classes = len(train_data.class_indices)

base = InceptionV3(
    include_top=False,
    weights="imagenet",
    input_shape=(224, 224, 3)
)

base.trainable = False

inputs = layers.Input(shape=(224, 224, 3))

# Apply InceptionV3 preprocessing
x = preprocess_input(inputs)

x = base(x, training=False)
x = layers.GlobalAveragePooling2D()(x)
x = layers.Dropout(0.4)(x)
outputs = layers.Dense(num_classes, activation="softmax")(x)

model_inception = models.Model(inputs, outputs)

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

#   Train
history_inception = model_inception.fit(
    train_data,
    validation_data=val_data,
    epochs=5
)

# Save model
model_inception.save("inceptionV3_sea_animals.keras")

from sklearn.metrics import (
    classification_report,
    accuracy_score,
    precision_score,
    recall_score,
    f1_score,
    confusion_matrix
)
#-----------------------------------
# Predictions
y_pred = model_inception.predict(test_data)
y_pred_classes = np.argmax(y_pred, axis=1)

# True labels
y_true = test_data.classes

# Class names
class_labels = list(test_data.class_indices.keys())

# Compute Metrics
accuracy = accuracy_score(y_true, y_pred_classes)
precision = precision_score(y_true, y_pred_classes, average='weighted')
recall = recall_score(y_true, y_pred_classes, average='weighted')
f1 = f1_score(y_true, y_pred_classes, average='weighted')

print("Accuracy:", accuracy)
print("Precision:", precision)
print("Recall:", recall)
print("F1-score:", f1)
```

```python
print("\nClassification Report:")
print(classification_report(y_true, y_pred_classes, target_names=class_labels))


cm = confusion_matrix(y_true, y_pred_classes)

plt.figure(figsize=(16, 12))
sns.heatmap(cm, annot=True, fmt="d", cmap="Purples",
            xticklabels=class_labels,
            yticklabels=class_labels)

plt.title("Confusion Matrix - InceptionV3", fontsize=18)
plt.ylabel("True Label", fontsize=14)
plt.xlabel("Predicted Label", fontsize=14)
plt.xticks(rotation=90)
plt.yticks(rotation=0)
plt.tight_layout()
plt.show()
```

```
Epoch 1/5
101/101 ━━━━━━━━━━━━━━━━━━━━ 118s 1s/step - accuracy: 0.4147 - loss: 1.4464 - val_accuracy: 0.4236 - val_loss: 1.3371
Epoch 2/5
101/101 ━━━━━━━━━━━━━━━━━━━━ 110s 1s/step - accuracy: 0.4541 - loss: 1.3277 - val_accuracy: 0.4454 - val_loss: 1.2877
Epoch 3/5
101/101 ━━━━━━━━━━━━━━━━━━━━ 113s 1s/step - accuracy: 0.4678 - loss: 1.3033 - val_accuracy: 0.4731 - val_loss: 1.2618
Epoch 4/5
101/101 ━━━━━━━━━━━━━━━━━━━━ 120s 1s/step - accuracy: 0.4852 - loss: 1.2827 - val_accuracy: 0.5007 - val_loss: 1.2217
Epoch 5/5
101/101 ━━━━━━━━━━━━━━━━━━━━ 126s 1s/step - accuracy: 0.4871 - loss: 1.2633 - val_accuracy: 0.4978 - val_loss: 1.2064
22/22 ━━━━━━━━━━━━━━━━━━━━ 25s 1s/step
Accuracy: 0.4899135446685879
Precision: 0.4533977428491987
Recall: 0.4899135446685879
F1-score: 0.39653034514255664

Classification Report:
                 precision    recall  f1-score   support

        Dolphin       0.34      0.38      0.36       118
     Jelly Fish       0.67      0.17      0.27       128
       Seahorse       0.57      0.05      0.10        73
         Sharks       0.00      0.00      0.00        89
Turtle_Tortoise       0.52      0.94      0.67       286

       accuracy                           0.49       694
      macro avg       0.42      0.31      0.28       694
   weighted avg       0.45      0.49      0.40       694
```

```
/opt/anaconda3/lib/python3.12/site-packages/sklearn/metrics/_classification.py:1531: UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 in labels wit
h no predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/opt/anaconda3/lib/python3.12/site-packages/sklearn/metrics/_classification.py:1531: UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 in labels wit
h no predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/opt/anaconda3/lib/python3.12/site-packages/sklearn/metrics/_classification.py:1531: UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 in labels wit
h no predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/opt/anaconda3/lib/python3.12/site-packages/sklearn/metrics/_classification.py:1531: UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 in labels wit
h no predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
```



In [ ]: