In [1]:
```
import warnings
warnings.filterwarnings('ignore')
```

In [2]:
```
from tensorflow.keras.layers import Dense, Flatten, Input, Lambda
from tensorflow.keras.models import Model ,Sequential
from tensorflow.keras.preprocessing import image
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Dense, GlobalAveragePooling2D
from tensorflow.keras.preprocessing.image import ImageDataGenerator, load_img
from glob import glob
import matplotlib.pyplot as plt
import numpy as np
import tensorflow as tf
```

In [3]:
```
import tensorflow as tf
import tensorflow_addons as tfa
from tensorflow import keras
from tensorflow.keras import layers
```

In [10]:
```

input_shape = (256, 256, 3)
num_classes = 4

train_path = 'Brain_Tumor_MRI_Image_Dataset/Training'
test_path = 'Brain_Tumor_MRI_Image_Dataset/Testing'
```

In [11]:
```
# Scaling all the images between 0 to 1

train_datagen = ImageDataGenerator(rescale = 1./255, shear_range=0.2, zoom_range=0.2, horizontal_flip=False)

# Performing only scaling on the test dataset

test_datagen = ImageDataGenerator(rescale=1./255)
```

In [12]:
```
train_set = train_datagen.flow_from_directory(train_path,
                                              target_size=(256,256),
                                              batch_size=2,
                                              class_mode = 'categorical')

test_set = test_datagen.flow_from_directory(test_path,
                                            target_size=(256,256),
                                            batch_size=2,
                                            class_mode='categorical')
```

```
Found 5712 images belonging to 4 classes.
Found 1311 images belonging to 4 classes.
```

In [3]:
```
from keras import backend as K

def recall_m(y_true, y_pred):
    true_positives = K.sum(K.round(K.clip(y_true * y_pred, 0, 1)))
    possible_positives = K.sum(K.round(K.clip(y_true, 0, 1)))
    recall = true_positives / (possible_positives + K.epsilon())
    return recall

def precision_m(y_true, y_pred):
    true_positives = K.sum(K.round(K.clip(y_true * y_pred, 0, 1)))
    predicted_positives = K.sum(K.round(K.clip(y_pred, 0, 1)))
    precision = true_positives / (predicted_positives + K.epsilon())
    return precision

def f1_m(y_true, y_pred):
    precision = precision_m(y_true, y_pred)
    recall = recall_m(y_true, y_pred)
    return 2*((precision*recall)/(precision+recall+K.epsilon()))
```

In [4]:
```
from tensorflow.keras.applications import Xception,NASNetLarge, NASNetMobile, VGG16,VGG19,InceptionV3,ResNet50,InceptionResNetV2
```

In [22]:
```
from keras.metrics import Precision, Recall
from keras.callbacks import EarlyStopping, ReduceLROnPlateau
from tensorflow.keras.models import Sequential, Model

learning_rate_reduction = ReduceLROnPlateau(
    monitor="val_accuracy", patience=3, verbose=1, factor=0.3, min_lr=0.0000001
)
early_stop = EarlyStopping(
    patience=10,
    verbose=1,
    monitor="val_accuracy",
    mode="max",
    min_delta=0.001,
    restore_best_weights=True,
)
```

In [65]:
```
ML_Model = []
accuracy = []
precision = []
recall = []
f1score = []

#function to call for storing the results
def storeResults(model, a,b,c,d):
    ML_Model.append(model)
    accuracy.append(round(a, 3))
    precision.append(round(b, 3))
    recall.append(round(c, 3))
    f1score.append(round(d, 3))
```

ViT Models

SWIN

In [16]:
```
patch_size = (2, 2)  # 2-by-2 sized patches
dropout_rate = 0.03  # Dropout rate
num_heads = 8  # Attention heads
embed_dim = 64  # Embedding dimension
num_mlp = 256  # MLP layer size
qkv_bias = True  # Convert embedded patches to query, key, and values with a learnable additive value
window_size = 2  # Size of attention window
shift_size = 1  # Size of shifting window
image_dimension = 256  # Initial image size

num_patch_x = input_shape[0] // patch_size[0]
num_patch_y = input_shape[1] // patch_size[1]

learning_rate = 1e-3
batch_size = 2
num_epochs = 40
validation_split = 0.1
weight_decay = 0.0001
label_smoothing = 0.1
```

In [17]:
```
def window_partition(x, window_size):
    _, height, width, channels = x.shape
    patch_num_y = height // window_size
    patch_num_x = width // window_size
    x = tf.reshape(
        x, shape=(-1, patch_num_y, window_size, patch_num_x, window_size, channels)
    )
    x = tf.transpose(x, (0, 1, 3, 2, 4, 5))
    windows = tf.reshape(x, shape=(-1, window_size, window_size, channels))
    return windows


def window_reverse(windows, window_size, height, width, channels):
    patch_num_y = height // window_size
    patch_num_x = width // window_size
    x = tf.reshape(
        windows,
        shape=(-1, patch_num_y, patch_num_x, window_size, window_size, channels),
    )
    x = tf.transpose(x, perm=(0, 1, 3, 2, 4, 5))
    x = tf.reshape(x, shape=(-1, height, width, channels))
    return x


class DropPath(layers.Layer):
    def __init__(self, drop_prob=None, **kwargs):
        super(DropPath, self).__init__(**kwargs)
        self.drop_prob = drop_prob

    def call(self, x):
        input_shape = tf.shape(x)
        batch_size = input_shape[0]
        rank = x.shape.rank
        shape = (batch_size,) + (1,) * (rank - 1)
        random_tensor = (1 - self.drop_prob) + tf.random.uniform(shape, dtype=x.dtype)
        path_mask = tf.floor(random_tensor)
        output = tf.math.divide(x, 1 - self.drop_prob) * path_mask
        return output
```

In [18]:
```
class WindowAttention(layers.Layer):
    def __init__(
        self, dim, window_size, num_heads, qkv_bias=True, dropout_rate=0.0, **kwargs
    ):
        super(WindowAttention, self).__init__(**kwargs)
        self.dim = dim
        self.window_size = window_size
        self.num_heads = num_heads
        self.scale = (dim // num_heads) ** -0.5
        self.qkv = layers.Dense(dim * 3, use_bias=qkv_bias)
        self.dropout = layers.Dropout(dropout_rate)
        self.proj = layers.Dense(dim)

    def build(self, input_shape):
        num_window_elements = (2 * self.window_size[0] - 1) * (
            2 * self.window_size[1] - 1
        )
        self.relative_position_bias_table = self.add_weight(
            shape=(num_window_elements, self.num_heads),
            initializer=tf.initializers.Zeros(),
            trainable=True,
        )
        coords_h = np.arange(self.window_size[0])
        coords_w = np.arange(self.window_size[1])
        coords_matrix = np.meshgrid(coords_h, coords_w, indexing="ij")
        coords = np.stack(coords_matrix)
        coords_flatten = coords.reshape(2, -1)
        relative_coords = coords_flatten[:, :, None] - coords_flatten[:, None, :]
        relative_coords = relative_coords.transpose([1, 2, 0])
        relative_coords[:, :, 0] += self.window_size[0] - 1
        relative_coords[:, :, 1] += self.window_size[1] - 1
        relative_coords[:, :, 0] *= 2 * self.window_size[1] - 1
        relative_position_index = relative_coords.sum(-1)

        self.relative_position_index = tf.Variable(
            initial_value=tf.convert_to_tensor(relative_position_index), trainable=False
        )

    def call(self, x, mask=None):
        _, size, channels = x.shape
        head_dim = channels // self.num_heads
        x_qkv = self.qkv(x)
        x_qkv = tf.reshape(x_qkv, shape=(-1, size, 3, self.num_heads, head_dim))
        x_qkv = tf.transpose(x_qkv, perm=(2, 0, 3, 1, 4))
        q, k, v = x_qkv[0], x_qkv[1], x_qkv[2]
        q = q * self.scale
        k = tf.transpose(k, perm=(0, 1, 3, 2))
        attn = q @ k

        num_window_elements = self.window_size[0] * self.window_size[1]
        relative_position_index_flat = tf.reshape(
            self.relative_position_index, shape=(-1,)
        )
        relative_position_bias = tf.gather(
            self.relative_position_bias_table, relative_position_index_flat
        )
        relative_position_bias = tf.reshape(
            relative_position_bias, shape=(num_window_elements, num_window_elements, -1)
        )
        relative_position_bias = tf.transpose(relative_position_bias, perm=(2, 0, 1))
        attn = attn + tf.expand_dims(relative_position_bias, axis=0)

        if mask is not None:
            nW = mask.get_shape()[0]
            mask_float = tf.cast(
                tf.expand_dims(tf.expand_dims(mask, axis=1), axis=0), tf.float32
            )
            attn = (
                tf.reshape(attn, shape=(-1, nW, self.num_heads, size, size))
                + mask_float
            )
            attn = tf.reshape(attn, shape=(-1, self.num_heads, size, size))
            attn = keras.activations.softmax(attn, axis=-1)
        else:
            attn = keras.activations.softmax(attn, axis=-1)
        attn = self.dropout(attn)

        x_qkv = attn @ v
        x_qkv = tf.transpose(x_qkv, perm=(0, 2, 1, 3))
        x_qkv = tf.reshape(x_qkv, shape=(-1, size, channels))
        x_qkv = self.proj(x_qkv)
        x_qkv = self.dropout(x_qkv)
        return x_qkv
```

In [19]:
```
class SwinTransformer(layers.Layer):
    def __init__(
        self,
        dim,
        num_patch,
        num_heads,
        window_size=7,
        shift_size=0,
        num_mlp=1024,
        qkv_bias=True,
        dropout_rate=0.0,
        **kwargs,
    ):
        super(SwinTransformer, self).__init__(**kwargs)

        self.dim = dim  # number of input dimensions
        self.num_patch = num_patch  # number of embedded patches
        self.num_heads = num_heads  # number of attention heads
        self.window_size = window_size  # size of window
        self.shift_size = shift_size  # size of window shift
        self.num_mlp = num_mlp  # number of MLP nodes

        self.norm1 = layers.LayerNormalization(epsilon=1e-5)
        self.attn = WindowAttention(
            dim,
            window_size=(self.window_size, self.window_size),
            num_heads=num_heads,
            qkv_bias=qkv_bias,
            dropout_rate=dropout_rate,
        )
        self.drop_path = DropPath(dropout_rate)
        self.norm2 = layers.LayerNormalization(epsilon=1e-5)

        self.mlp = keras.Sequential(
            [
                layers.Dense(num_mlp),
                layers.Activation(keras.activations.gelu),
                layers.Dropout(dropout_rate),
                layers.Dense(dim),
                layers.Dropout(dropout_rate),
            ]
        )

        if min(self.num_patch) < self.window_size:
            self.shift_size = 0
            self.window_size = min(self.num_patch)

    def build(self, input_shape):
        if self.shift_size == 0:
            self.attn_mask = None
        else:
            height, width = self.num_patch
            h_slices = (
                slice(0, -self.window_size),
                slice(-self.window_size, -self.shift_size),
                slice(-self.shift_size, None),
            )
            w_slices = (
                slice(0, -self.window_size),
                slice(-self.window_size, -self.shift_size),
                slice(-self.shift_size, None),
            )
            mask_array = np.zeros((1, height, width, 1))
            count = 0
            for h in h_slices:
                for w in w_slices:
                    mask_array[:, h, w, :] = count
                    count += 1
            mask_array = tf.convert_to_tensor(mask_array)

            # mask array to windows
            mask_windows = window_partition(mask_array, self.window_size)
            mask_windows = tf.reshape(
                mask_windows, shape=[-1, self.window_size * self.window_size]
            )
            attn_mask = tf.expand_dims(mask_windows, axis=1) - tf.expand_dims(
                mask_windows, axis=2
            )
            attn_mask = tf.where(attn_mask != 0, -100.0, attn_mask)
            attn_mask = tf.where(attn_mask == 0, 0.0, attn_mask)
            self.attn_mask = tf.Variable(initial_value=attn_mask, trainable=False)

    def call(self, x):
        height, width = self.num_patch
        _, num_patches_before, channels = x.shape
        x_skip = x
        x = self.norm1(x)
        x = tf.reshape(x, shape=(-1, height, width, channels))
        if self.shift_size > 0:
            shifted_x = tf.roll(
                x, shift=[-self.shift_size, -self.shift_size], axis=[1, 2]
            )
        else:
            shifted_x = x

        x_windows = window_partition(shifted_x, self.window_size)
        x_windows = tf.reshape(
            x_windows, shape=(-1, self.window_size * self.window_size, channels)
        )
        attn_windows = self.attn(x_windows, mask=self.attn_mask)

        attn_windows = tf.reshape(
            attn_windows, shape=(-1, self.window_size, self.window_size, channels)
        )
        shifted_x = window_reverse(
            attn_windows, self.window_size, height, width, channels
        )
        if self.shift_size > 0:
            x = tf.roll(
                shifted_x, shift=[self.shift_size, self.shift_size], axis=[1, 2]
            )
        else:
            x = shifted_x

        x = tf.reshape(x, shape=(-1, height * width, channels))
        x = self.drop_path(x)
        x = x_skip + x
        x_skip = x
        x = self.norm2(x)
        x = self.mlp(x)
        x = self.drop_path(x)
        x = x_skip + x
        return x
```

In [20]:
```
class PatchExtract(layers.Layer):
    def __init__(self, patch_size, **kwargs):
        super(PatchExtract, self).__init__(**kwargs)
        self.patch_size_x = patch_size[0]
        self.patch_size_y = patch_size[0]

    def call(self, images):
        batch_size = tf.shape(images)[0]
        patches = tf.image.extract_patches(
            images=images,
            sizes=(1, self.patch_size_x, self.patch_size_y, 1),
            strides=(1, self.patch_size_x, self.patch_size_y, 1),
            rates=(1, 1, 1, 1),
            padding="VALID",
        )
        patch_dim = patches.shape[-1]
        patch_num = patches.shape[1]
        return tf.reshape(patches, (batch_size, patch_num * patch_num, patch_dim))


class PatchEmbedding(layers.Layer):
    def __init__(self, num_patch, embed_dim, **kwargs):
        super(PatchEmbedding, self).__init__(**kwargs)
        self.num_patch = num_patch
        self.proj = layers.Dense(embed_dim)
        self.pos_embed = layers.Embedding(input_dim=num_patch, output_dim=embed_dim)

    def call(self, patch):
        pos = tf.range(start=0, limit=self.num_patch, delta=1)
        return self.proj(patch) + self.pos_embed(pos)


class PatchMerging(tf.keras.layers.Layer):
    def __init__(self, num_patch, embed_dim):
        super(PatchMerging, self).__init__()
        self.num_patch = num_patch
        self.embed_dim = embed_dim
        self.linear_trans = layers.Dense(2 * embed_dim, use_bias=False)

    def call(self, x):
        height, width = self.num_patch
        _, _, C = x.get_shape().as_list()
        x = tf.reshape(x, shape=(-1, height, width, C))
        x0 = x[:, 0::2, 0::2, :]
        x1 = x[:, 1::2, 0::2, :]
        x2 = x[:, 0::2, 1::2, :]
        x3 = x[:, 1::2, 1::2, :]
        x = tf.concat((x0, x1, x2, x3), axis=-1)
        x = tf.reshape(x, shape=(-1, (height // 2) * (width // 2), 4 * C))
        return self.linear_trans(x)
```

In [21]:
```
input = layers.Input(input_shape)
x = layers.RandomCrop(image_dimension, image_dimension)(input)
x = layers.RandomFlip("horizontal")(x)
x = PatchExtract(patch_size)(x)
x = PatchEmbedding(num_patch_x * num_patch_y, embed_dim)(x)
x = SwinTransformer(
    dim=embed_dim,
    num_patch=(num_patch_x, num_patch_y),
    num_heads=num_heads,
    window_size=window_size,
    shift_size=0,
    num_mlp=num_mlp,
    qkv_bias=qkv_bias,
    dropout_rate=dropout_rate,
)(x)
x = SwinTransformer(
    dim=embed_dim,
    num_patch=(num_patch_x, num_patch_y),
    num_heads=num_heads,
    window_size=window_size,
    shift_size=shift_size,
    num_mlp=num_mlp,
    qkv_bias=qkv_bias,
    dropout_rate=dropout_rate,
)(x)
x = PatchMerging((num_patch_x, num_patch_y), embed_dim=embed_dim)(x)
x = layers.GlobalAveragePooling1D()(x)
output = layers.Dense(num_classes, activation="softmax")(x)
```

```
WARNING:tensorflow:Using a while_loop for converting RngReadAndSkip cause there is no registered converter for this op.
WARNING:tensorflow:Using a while_loop for converting Bitcast cause there is no registered converter for this op.
WARNING:tensorflow:Using a while_loop for converting Bitcast cause there is no registered converter for this op.
WARNING:tensorflow:Using a while_loop for converting StatelessRandomUniformIntV2 cause there is no registered converter for this op.
```

In [22]:
```
model1 = keras.Model(input, output)
model1.compile(loss = 'categorical_crossentropy', optimizer='adam', metrics=["accuracy",f1_m,precision_m, recall_m])
model1.summary()
```

```
Model: "model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 input_1 (InputLayer)        [(None, 256, 256, 3)]     0         
                                                                 
 random_crop (RandomCrop)    (None, 256, 256, 3)       0         
                                                                 
 random_flip (RandomFlip)    (None, 256, 256, 3)       0         
                                                                 
 patch_extract (PatchExtract  (None, 16384, 12)        0         
 )                                                               
                                                                 
 patch_embedding (PatchEmbed  (None, 16384, 64)        1049408   
 ding)                                                           
                                                                 
 swin_transformer (SwinTrans  (None, 16384, 64)        50072     
 former)                                                         
                                                                 
 swin_transformer_1 (SwinTra  (None, 16384, 64)        115608    
 nsformer)                                                       
                                                                 
 patch_merging (PatchMerging  (None, 4096, 128)        32768     
 )                                                               
                                                                 
 global_average_pooling1d (G  (None, 128)              0         
 lobalAveragePooling1D)                                          
                                                                 
 dense_10 (Dense)            (None, 4)                 516       
                                                                 
=================================================================
Total params: 1,248,372
Trainable params: 1,182,804
Non-trainable params: 65,568
_________________________________________________________________
```

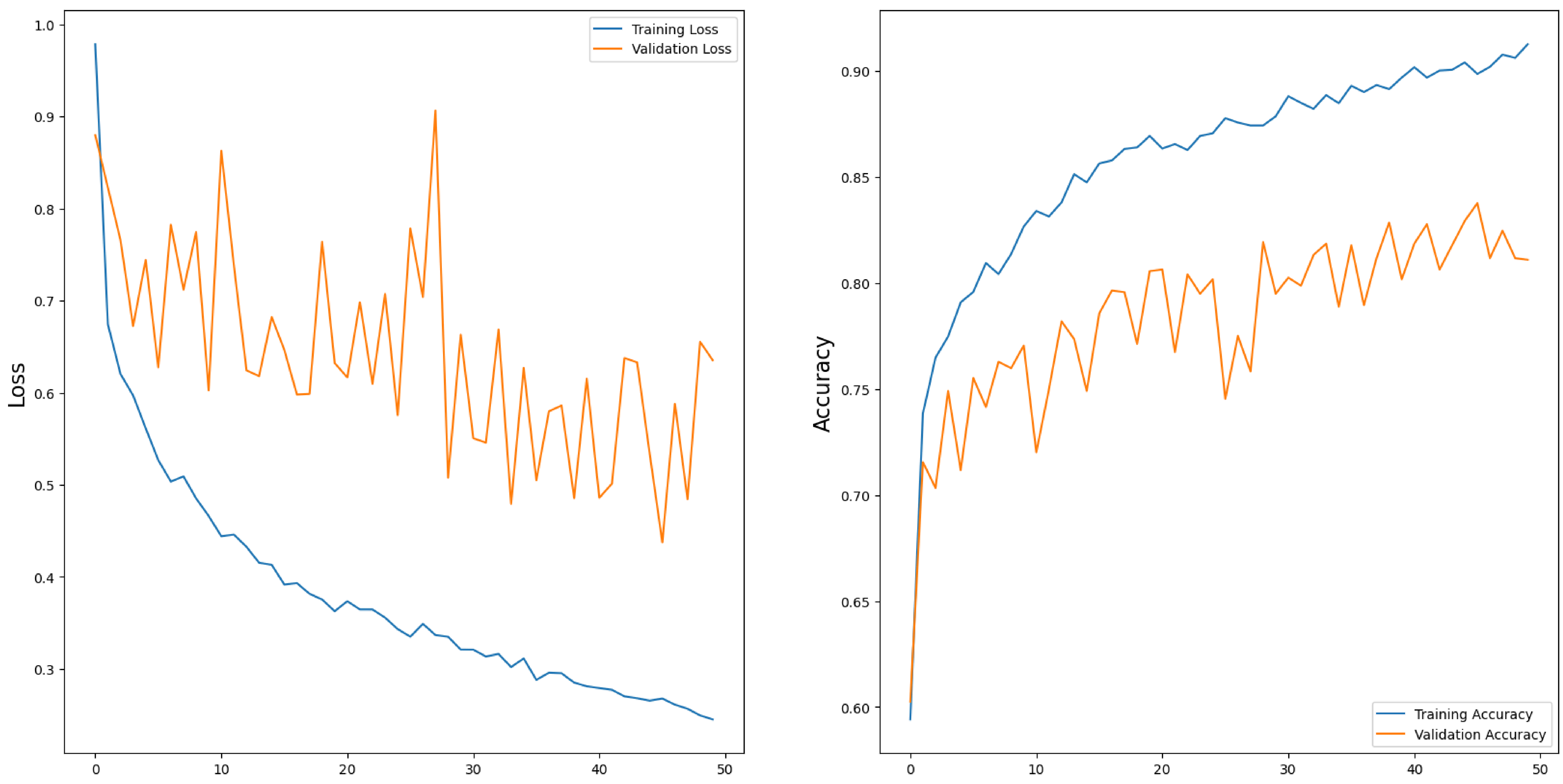
In [24]:
```
hist1 = model1.fit(train_set, validation_data=test_set, epochs=50, steps_per_epoch=len(train_set), validation_steps=len(test_set))#,callbacks=[learning_rate_reduction, early_stop]
```

```
Epoch 1/50
WARNING:tensorflow:Using a while_loop for converting RngReadAndSkip cause there is no registered converter for this op.
WARNING:tensorflow:Using a while_loop for converting Bitcast cause there is no registered converter for this op.
WARNING:tensorflow:Using a while_loop for converting Bitcast cause there is no registered converter for this op.
WARNING:tensorflow:Using a while_loop for converting StatelessRandomUniformIntV2 cause there is no registered converter for this op.
WARNING:tensorflow:Using a while_loop for converting RngReadAndSkip cause there is no registered converter for this op.
WARNING:tensorflow:Using a while_loop for converting Bitcast cause there is no registered converter for this op.
WARNING:tensorflow:Using a while_loop for converting Bitcast cause there is no registered converter for this op.
WARNING:tensorflow:Using a while_loop for converting StatelessRandomUniformIntV2 cause there is no registered converter for this op.
2856/2856 [==============================] - 134s 44ms/step - loss: 0.9785 - accuracy: 0.5942 - f1_m: 0.5037 - precision_m: 0.5921 - recall_m: 0.4596 - val_loss: 0.8797 - val_
accuracy: 0.6026 - val_f1_m: 0.5401 - val_precision_m: 0.6250 - val_recall_m: 0.4977
Epoch 2/50
2856/2856 [==============================] - 108s 38ms/step - loss: 0.6742 - accuracy: 0.7384 - f1_m: 0.7082 - precision_m: 0.7719 - recall_m: 0.6763 - val_loss: 0.8228 - val_
accuracy: 0.7155 - val_f1_m: 0.6880 - val_precision_m: 0.7363 - val_recall_m: 0.6639
Epoch 3/50
2856/2856 [==============================] - 109s 38ms/step - loss: 0.6203 - accuracy: 0.7647 - f1_m: 0.7382 - precision_m: 0.7971 - recall_m: 0.7087 - val_loss: 0.7658 - val_
accuracy: 0.7033 - val_f1_m: 0.6641 - val_precision_m: 0.7134 - val_recall_m: 0.6395
Epoch 4/50
2856/2856 [==============================] - 108s 38ms/step - loss: 0.5970 - accuracy: 0.7747 - f1_m: 0.7545 - precision_m: 0.8104 - recall_m: 0.7265 - val_loss: 0.6724 - val_
```

In [66]:
```
dl_acc = hist1.history["val_accuracy"][49]
dl_prec = hist1.history["val_precision_m"][49]
dl_rec = hist1.history["val_recall_m"][49]
dl_f1 = hist1.history["val_f1_m"][49]

storeResults('VisionTransformer - SWIN',dl_acc,dl_prec,dl_rec,dl_f1)
```

In [25]:
```
import matplotlib.pyplot as plt

x=hist1
plt.figure(figsize=(20,10))
plt.subplot(1, 2, 1)
plt.suptitle('Optimizer : adam', fontsize=10)
plt.ylabel('Loss', fontsize=16)
plt.plot(x.history['loss'], label='Training Loss')
plt.plot(x.history['val_loss'], label='Validation Loss')
plt.legend(loc='upper right')

plt.subplot(1, 2, 2)
plt.ylabel('Accuracy', fontsize=16)
plt.plot(x.history['accuracy'], label='Training Accuracy')
plt.plot(x.history['val_accuracy'], label='Validation Accuracy')
plt.legend(loc='lower right')
plt.show()
```

Optimizer : adam



CCT

In [26]:
```
# Model
from keras.layers import add
from keras.layers import Dense
from keras.layers import Layer
from keras.layers import Input
from keras.layers import Conv2D
from keras.layers import Dropout
from keras.layers import Embedding
from keras.layers import MaxPooling2D as  MP
from keras.layers import ZeroPadding2D as  ZP
from keras.layers import LayerNormalization as LN
from keras.layers import MultiHeadAttention as MHA
from keras.models import Sequential, Model

# Optimizer
from tensorflow_addons.optimizers import AdamW

# Model Visualization
from tensorflow.keras.utils import plot_model
```

In [27]:
```
Learning_Rate  = 1e-3
Weight_Decay   = 1e-4
FILTERS        = [64,256]
Drop_Rates     = [0.0,0.1]
Dims           = 256
Hidden_units   = [Dims, Dims]
IMAGE_SIZE     = 256
```

In [28]:
```
def show_images(data, class_names, model=None, GRID=[5,6], SIZE=(30,25)):

    # Plotiing Configuration
    n_rows, n_cols = GRID
    n_images = n_rows * n_cols
    plt.figure(figsize=SIZE)

    # iterate through the data
    i=1
    for images, labels in iter(data):

        # Select data at random
        id = np.random.randint(len(images))
        image, label = tf.expand_dims(images[id],axis=0), class_names[int(labels[id])]

        # Make Prediciton
        if model is not None :
            prediction = model.predict(image)[0]
            score = np.round(np.max(prediction),2)
            pred  = class_names[np.argmax(prediction)]

            title = "True : {}\nPred : {}\n Score : {:.2}".format(label, pred, score)

        else:
            title = label
        cls()

        # Plot Image
        plt.subplot(n_rows, n_cols, i)
        plt.imshow(image[0])
        plt.axis('off')
        plt.title(title)

        # Break Loop
        i+=1
        if i>n_images: break

    # Show Final Figure
    plt.show()
```

In [29]:
```
# get the Class Names
import os
class_names = sorted(os.listdir(train_path))
n_classes   = len(class_names)

# Show
print("No. of Classes : {}\nClass Names : {}".format(n_classes, class_names))
```

```
No. of Classes : 4
Class Names : ['glioma', 'meningioma', 'notumor', 'pituitary']
```

In [30]:
```
class ConvTokenizer(Layer):

    def __init__(self, filters, **kwargs):
        super(ConvTokenizer, self).__init__(**kwargs)

        self.net = Sequential([
            Conv2D(filters[0], kernel_size=3, strides=1, padding='valid', activation='relu', kernel_initializer='he_normal', use_bias=False, name="TokenizerConv1"),
            ZP(padding=1, name="ZP1"),
            MP(pool_size=3, strides=2, padding='same', name="MP1"),
            Conv2D(filters[1], kernel_size=3, strides=1, padding='valid', activation='relu', kernel_initializer='he_normal', use_bias=False, name="TokenizerConv2"),
            ZP(padding=1, name="ZP2"),
            MP(pool_size=3, strides=2, padding='same', name="MP2"),
        ], name="ConvTokenizer")

        self.filters = filters

    def call(self, X):
        patches = self.net(X)
        patches = tf.reshape(patches, shape=(-1, patches.shape[1]*patches.shape[2], patches.shape[-1]))
        return patches

    def PE(self, IMAGE_SIZE):
        dummy_inputs  = tf.ones(shape=(1, IMAGE_SIZE, IMAGE_SIZE, 3))
        dummy_outputs = self.call(dummy_inputs)

        seq_len  = tf.shape(dummy_outputs)[1]
        proj_dim = tf.shape(dummy_outputs)[-1]
        EL = Embedding(input_dim=seq_len, output_dim=proj_dim, name="PositionalEmbedding")
        return EL, seq_len

    def get_config(self):
        base_config = super().get_config()
        return {**base_config,"filters":self.filters}
```

In [31]:
```
def SeqPool(x):

    rep = LN(epsilon=1e-5, name="LayerNormSeqPool")(x)
    y   = Dense(1, name="LinearTransformation")(rep)

    attention_weights = tf.nn.softmax(y, axis=-1)
    weighted_rep = tf.squeeze(tf.matmul(attention_weights, rep, transpose_a=True),axis=-2)
    return weighted_rep
```

In [32]:
```
class MLPBlock(Layer):

    def __init__(self, units, rate, **kwargs):
        super(MLPBlock, self).__init__(**kwargs)

        self.units = units
        self.rate = rate

        self.net = Sequential(name='MLPBlock')
        for unit in units:
            self.net.add(Dense(unit, activation='relu', kernel_initializer='he_normal'))
            self.net.add(Dropout(rate))

    def call(self, X):
        return self.net(X)

    def get_config(self):
        base_config = super().get_config()
        return {
            **base_config,
            "units":self.units,
            "rate":self.rate
        }
```

In [33]:
```
class StochasticDepthRegularization(Layer):

    def __init__(self, drop_rate, **kwargs):
        super(StochasticDepthRegularization, self).__init__(**kwargs)

        self.dr = drop_rate

    def call(self, x, training=None):
        if training:
            keep_rate = 1 - self.dr
            shape = (tf.shape(x)[0],) + (1,) * (tf.shape(x).shape[0] - 1)
            random_tensor = keep_rate + tf.random.uniform(shape, 0, 1)
            random_tensor = tf.floor(random_tensor)
            return (x / keep_rate) * random_tensor
        return x
```

In [34]:
```
def TransformerBlock(x, rates=Drop_Rates, L=2, proj_dim=Dims):
    for i in range(L):
        y = LN(epsilon=1e-5, name="LayerNormTB_Lower_{}".format(i+1))(x)
        y = MHA(num_heads=2, key_dim=Dims, dropout=0.1, name="MHA_{}".format(i+1))(y, y)
        y = StochasticDepthRegularization(drop_rate=rates[i],name="SDR_Lower_{}".format(i+1))(y)

        skip = add([y, x], name="SkipConnection_Lower_{}".format(i+1))

        y = LN(epsilon=1e-5, name="LayerNormTB_Top_{}".format(i+1))(skip)
        y = MLPBlock(Hidden_units, 0.1, name="MLPBlock_TB_{}".format(i+1))(y)
        y = StochasticDepthRegularization(drop_rate=rates[i],name="SDR_Top_{}".format(i+1))(y)

        x = add([y, skip], name="SkipConnection_Top_{}".format(i+1))
    return x
```

In [35]:
```
# Input Layer
InputL = Input(shape=(IMAGE_SIZE, IMAGE_SIZE, 3), name="InputLayer")

# Tokenizer
Tokenizer = ConvTokenizer(filters=FILTERS, name="ConvTokenizer")
tokens    = Tokenizer(InputL)


# Positional Embeddings
EL, seq_len = Tokenizer.PE(IMAGE_SIZE)
positions   = tf.range(start=0, limit=seq_len, delta=1)
embedding   = EL(positions)
tokens     += embedding

# Transformer Block
encodings = TransformerBlock(tokens)

# Sequence Pooling
pooled = SeqPool(encodings)

# Classifier
OutputL = Dense(n_classes, activation='softmax', name="OutptLayer")(pooled)

# Model
model2 = Model(InputL, OutputL, name="CCT")
model2.summary()
```

```
Model: "CCT"
__________________________________________________________________________________________________
 Layer (type)                   Output Shape         Param #     Connected to                     
==================================================================================================
 InputLayer (InputLayer)        [(None, 256, 256, 3  0           []                               
                                )]                                                                
                                                                                                  
 ConvTokenizer (ConvTokenizer)  (None, 4096, 256)    149184      ['InputLayer[0][0]']             
                                                                                                  
 tf.__operators__.add (TFOpLamb  (None, 4096, 256)   0           ['ConvTokenizer[0][0]']          
 da)                                                                                              
                                                                                                  
 LayerNormTB_Lower_1 (LayerNorm  (None, 4096, 256)   512         ['tf.__operators__.add[0][0]']   
 alization)                                                                                       
                                                                                                  
 MHA_1 (MultiHeadAttention)     (None, 4096, 256)    526080      ['LayerNormTB_Lower_1[0][0]',    
                                                                  'LayerNormTB_Lower_1[0][0]']    
                                                                                                  
 SDR_Lower_1 (StochasticDepthRe  (None, 4096, 256)   0           ['MHA_1[0][0]']                  
 gularization)                                                                                    
                                                                                                  
 SkipConnection_Lower_1 (Add)   (None, 4096, 256)    0           ['SDR_Lower_1[0][0]',            
                                                                  'tf.__operators__.add[0][0]']   
                                                                                                  
 LayerNormTB_Top_1 (LayerNormal  (None, 4096, 256)   512         ['SkipConnection_Lower_1[0][0]'] 
 ization)                                                                                         
                                                                                                  
 MLPBlock_TB_1 (MLPBlock)       (None, 4096, 256)    131584      ['LayerNormTB_Top_1[0][0]']      
                                                                                                  
 SDR_Top_1 (StochasticDepthRegu  (None, 4096, 256)   0           ['MLPBlock_TB_1[0][0]']          
 larization)                                                                                      
                                                                                                  
 SkipConnection_Top_1 (Add)     (None, 4096, 256)    0           ['SDR_Top_1[0][0]',              
                                                                  'SkipConnection_Lower_1[0][0]'] 
                                                                                                  
 LayerNormTB_Lower_2 (LayerNorm  (None, 4096, 256)   512         ['SkipConnection_Top_1[0][0]']   
 alization)                                                                                       
                                                                                                  
 MHA_2 (MultiHeadAttention)     (None, 4096, 256)    526080      ['LayerNormTB_Lower_2[0][0]',    
                                                                  'LayerNormTB_Lower_2[0][0]']    
                                                                                                  
 SDR_Lower_2 (StochasticDepthRe  (None, 4096, 256)   0           ['MHA_2[0][0]']                  
 gularization)                                                                                    
                                                                                                  
 SkipConnection_Lower_2 (Add)   (None, 4096, 256)    0           ['SDR_Lower_2[0][0]',            
                                                                  'SkipConnection_Top_1[0][0]']   
                                                                                                  
 LayerNormTB_Top_2 (LayerNormal  (None, 4096, 256)   512         ['SkipConnection_Lower_2[0][0]'] 
 ization)                                                                                         
                                                                                                  
 MLPBlock_TB_2 (MLPBlock)       (None, 4096, 256)    131584      ['LayerNormTB_Top_2[0][0]']      
                                                                                                  
 SDR_Top_2 (StochasticDepthRegu  (None, 4096, 256)   0           ['MLPBlock_TB_2[0][0]']          
 larization)                                                                                      
                                                                                                  
 SkipConnection_Top_2 (Add)     (None, 4096, 256)    0           ['SDR_Top_2[0][0]',              
                                                                  'SkipConnection_Lower_2[0][0]'] 
                                                                                                  
 LayerNormSeqPool (LayerNormali  (None, 4096, 256)   512         ['SkipConnection_Top_2[0][0]']   
 zation)                                                                                          
                                                                                                  
 LinearTransformation (Dense)   (None, 4096, 1)      257         ['LayerNormSeqPool[0][0]']       
                                                                                                  
 tf.nn.softmax (TFOpLambda)     (None, 4096, 1)      0           ['LinearTransformation[0][0]']   
                                                                                                  
 tf.linalg.matmul (TFOpLambda)  (None, 1, 256)       0           ['tf.nn.softmax[0][0]',          
                                                                  'LayerNormSeqPool[0][0]']       
                                                                                                  
 tf.compat.v1.squeeze (TFOpLamb  (None, 256)         0           ['tf.linalg.matmul[0][0]']       
 da)                                                                                              
                                                                                                  
 OutptLayer (Dense)             (None, 4)            1028        ['tf.compat.v1.squeeze[0][0]']   
                                                                                                  
==================================================================================================
Total params: 1,468,357
Trainable params: 1,468,357
Non-trainable params: 0
__________________________________________________________________________________________________
```

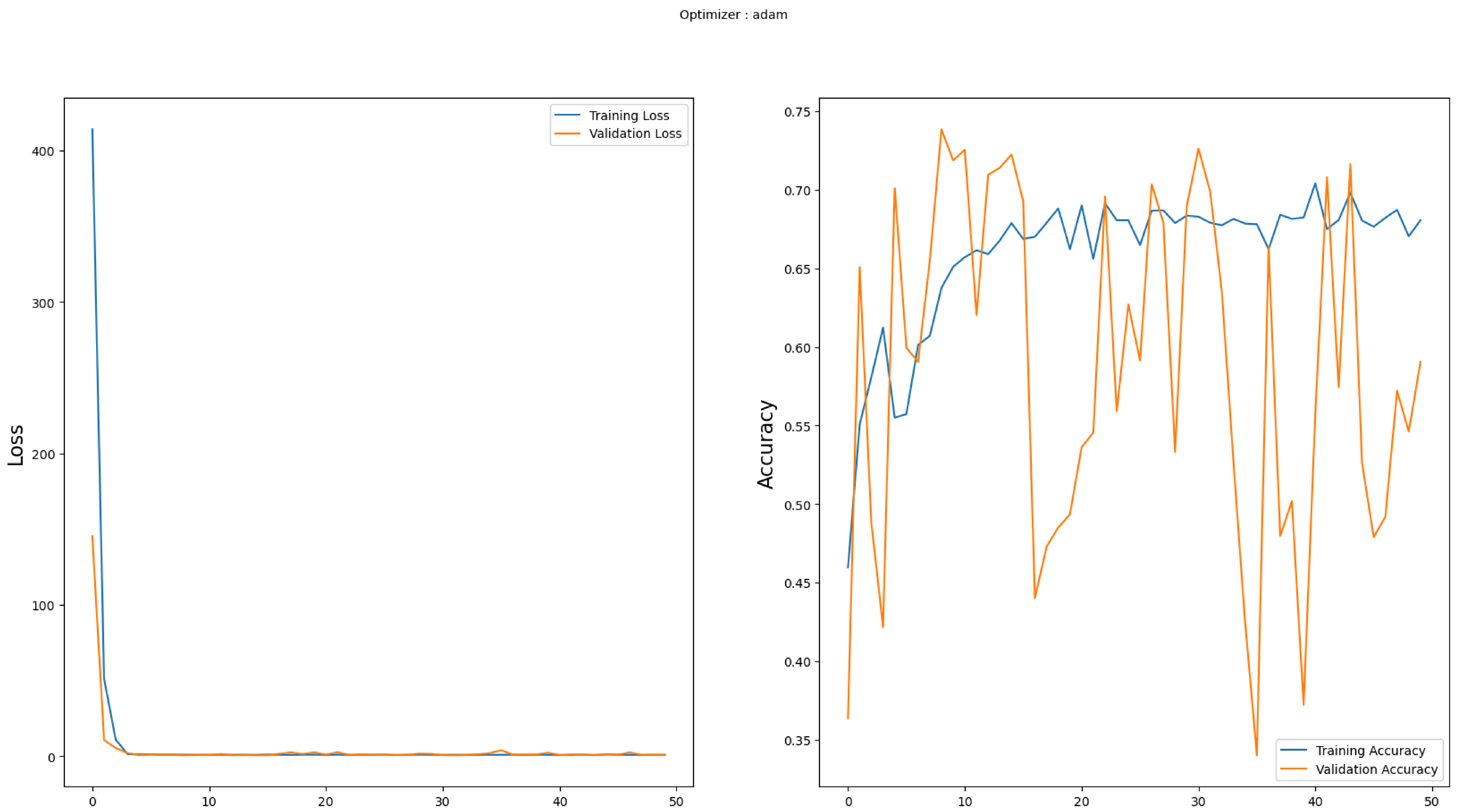
In [36]:
```
opt = AdamW(learning_rate=Learning_Rate, weight_decay=Weight_Decay)
model2.compile(loss = 'categorical_crossentropy', optimizer=opt, metrics=["accuracy",f1_m,precision_m, recall_m])
```

In [37]:
```
hist2 = model2.fit(train_set, validation_data=test_set, epochs=50, steps_per_epoch=len(train_set), validation_steps=len(test_set))#,callbacks=[learning_rate_reduction, early_stop
```

```
Epoch 1/50
2856/2856 [==============================] - 127s 41ms/step - loss: 414.0080 - accuracy: 0.4597 - f1_m: 0.4597 - precision_m: 0.4597 - recall_m: 0.4597 - val_loss: 145.4632 - val_accuracy: 0.3638 - val_f1_m: 0.3643 - val_precision_m: 0.3643 - val_recall_m: 0.3643
Epoch 2/50
2856/2856 [==============================] - 117s 41ms/step - loss: 51.3703 - accuracy: 0.5506 - f1_m: 0.5507 - precision_m: 0.5508 - recall_m: 0.5506 - val_loss: 10.8244 - val_accuracy: 0.6506 - val_f1_m: 0.6512 - val_precision_m: 0.6532 - val_recall_m: 0.6502
Epoch 3/50
2856/2856 [==============================] - 117s 41ms/step - loss: 11.0667 - accuracy: 0.5807 - f1_m: 0.5809 - precision_m: 0.5819 - recall_m: 0.5804 - val_loss: 5.5326 - val_accuracy: 0.4882 - val_f1_m: 0.4886 - val_precision_m: 0.4886 - val_recall_m: 0.4886
Epoch 4/50
2856/2856 [==============================] - 117s 41ms/step - loss: 1.5594 - accuracy: 0.6122 - f1_m: 0.5961 - precision_m: 0.6285 - recall_m: 0.5798 - val_loss: 2.2409 - val_accuracy: 0.4218 - val_f1_m: 0.4009 - val_precision_m: 0.4284 - val_recall_m: 0.3872
Epoch 5/50
2856/2856 [==============================] - 117s 41ms/step - loss: 1.3099 - accuracy: 0.5550 - f1_m: 0.5244 - precision_m: 0.5772 - recall_m: 0.4981 - val_loss: 0.8488 - val_accuracy: 0.7010 - val_f1_m: 0.5526 - val_precision_m: 0.6806 - val_recall_m: 0.4886
Epoch 6/50
2856/2856 [==============================] - 117s 41ms/step - loss: 1.3225 - accuracy: 0.5572 - f1_m: 0.5275 - precision_m: 0.5776 - recall_m: 0.5025 - val_loss: 1.1885 - val_accuracy: 0.5995 - val_f1_m: 0.5498 - val_precision_m: 0.6296 - val_recall_m: 0.5099
Epoch 7/50
```

In [67]:
```
dl_acc = hist2.history["val_accuracy"][49]
dl_prec = hist2.history["val_precision_m"][49]
dl_rec = hist2.history["val_recall_m"][49]
dl_f1 = hist2.history["val_f1_m"][49]

storeResults('VisionTransformer - CCT',dl_acc,dl_prec,dl_rec,dl_f1)
```

In [38]:
```
x=hist2
plt.figure(figsize=(20,10))
plt.subplot(1, 2, 1)
plt.suptitle('Optimizer : adam', fontsize=10)
plt.ylabel('Loss', fontsize=16)
plt.plot(x.history['loss'], label='Training Loss')
plt.plot(x.history['val_loss'], label='Validation Loss')
plt.legend(loc='upper right')

plt.subplot(1, 2, 2)
plt.ylabel('Accuracy', fontsize=16)
plt.plot(x.history['accuracy'], label='Training Accuracy')
plt.plot(x.history['val_accuracy'], label='Validation Accuracy')
plt.legend(loc='lower right')
plt.show()
```



EANet

In [39]:
```
import keras
from keras import layers
```

In [40]:
```
learning_rate = 0.001
weight_decay = 0.0001
batch_size = 128
num_epochs = 10
```

In [41]:
```
image_size = 256
auto = tf.data.AUTOTUNE

augmentation_layers = [
    keras.layers.RandomCrop(256, 256),
    keras.layers.RandomFlip("horizontal"),
]
```

In [42]:
```python
def activation_block(x):
    x = layers.Activation("gelu")(x)
    return layers.BatchNormalization()(x)


def conv_stem(x, filters: int, patch_size: int):
    x = layers.Conv2D(filters, kernel_size=patch_size, strides=patch_size)(x)
    return activation_block(x)


def conv_mixer_block(x, filters: int, kernel_size: int):
    # Depthwise convolution.
    x0 = x
    x = layers.DepthwiseConv2D(kernel_size=kernel_size, padding="same")(x)
    x = layers.Add()([activation_block(x), x0])  # Residual.

    # Pointwise convolution.
    x = layers.Conv2D(filters, kernel_size=1)(x)
    x = activation_block(x)

    return x


def eanet(
    image_size=32, filters=768, depth=8, kernel_size=5, patch_size=2, num_classes=7
):

    inputs = keras.Input((image_size, image_size, 3))
    x = layers.Rescaling(scale=1.0 / 255)(inputs)

    # Extract patch embeddings.
    x = conv_stem(x, filters, patch_size)

    # ConvMixer blocks.
    for _ in range(depth):
        x = conv_mixer_block(x, filters, kernel_size)

    # Classification block.
    x = layers.GlobalAvgPool2D()(x)
    outputs = layers.Dense(4, activation="softmax")(x)

    return keras.Model(inputs, outputs)
```

In [43]:
```python
model2 = eanet()
```

In [44]:
```python
model2.compile(loss = 'categorical_crossentropy', optimizer='adam', metrics=["accuracy",f1_m,precision_m, recall_m])
model2.summary()
```

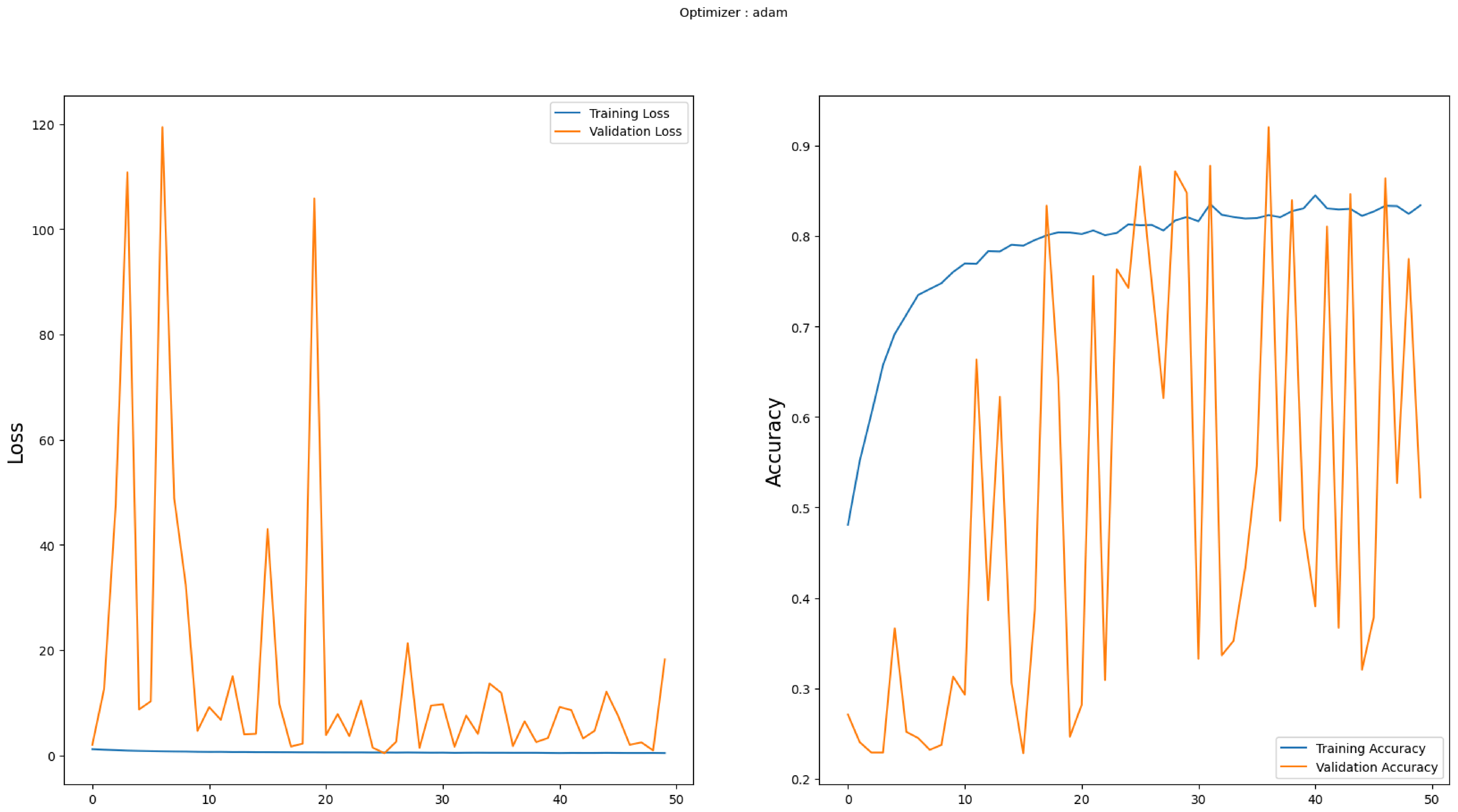
```
Model: "model_1"
__________________________________________________________________________________________________
 Layer (type)                   Output Shape         Param #     Connected to
==================================================================================================
 input_2 (InputLayer)           [(None, 32, 32, 3)]  0           []

 rescaling (Rescaling)          (None, 32, 32, 3)    0           ['input_2[0][0]']

 conv2d (Conv2D)                (None, 16, 16, 768)  9984        ['rescaling[0][0]']

 activation_2 (Activation)      (None, 16, 16, 768)  0           ['conv2d[0][0]']

 batch_normalization (BatchNorm  (None, 16, 16, 768)  3072       ['activation_2[0][0]']
 alization)

 depthwise_conv2d (DepthwiseCon  (None, 16, 16, 768)  19968      ['batch_normalization[0][0]']
 v2D)

 activation_3 (Activation)      (None, 16, 16, 768)  0           ['depthwise_conv2d[0][0]']
```

In [45]:
```python
hist2a = model2.fit(train_set, validation_data=test_set, epochs=50, steps_per_epoch=len(train_set), validation_steps=len(test_set))#,callbacks=[learning_rate_reduction, early_stop
```

```
Epoch 1/50
2856/2856 [==============================] - 496s 172ms/step - loss: 1.1885 - accuracy: 0.4807 - f1_m: 0.2741 - precision_m: 0.3528 - recall_m: 0.2348 - val_loss: 2.0107 - val
_accuracy: 0.2708 - val_f1_m: 0.2612 - val_precision_m: 0.2851 - val_recall_m: 0.2492
Epoch 2/50
2856/2856 [==============================] - 492s 172ms/step - loss: 1.0829 - accuracy: 0.5513 - f1_m: 0.3846 - precision_m: 0.4795 - recall_m: 0.3372 - val_loss: 12.5834 - va
l_accuracy: 0.2403 - val_f1_m: 0.2411 - val_precision_m: 0.2416 - val_recall_m: 0.2409
Epoch 3/50
2856/2856 [==============================] - 492s 172ms/step - loss: 0.9928 - accuracy: 0.6033 - f1_m: 0.4950 - precision_m: 0.5917 - recall_m: 0.4466 - val_loss: 47.5491 - va
l_accuracy: 0.2288 - val_f1_m: 0.2287 - val_precision_m: 0.2287 - val_recall_m: 0.2287
Epoch 4/50
2856/2856 [==============================] - 492s 172ms/step - loss: 0.9022 - accuracy: 0.6576 - f1_m: 0.5778 - precision_m: 0.6637 - recall_m: 0.5348 - val_loss: 110.7900 - v
al_accuracy: 0.2288 - val_f1_m: 0.2294 - val_precision_m: 0.2294 - val_recall_m: 0.2294
Epoch 5/50
2856/2856 [==============================] - 492s 172ms/step - loss: 0.8473 - accuracy: 0.6919 - f1_m: 0.6257 - precision_m: 0.7034 - recall_m: 0.5868 - val_loss: 8.7278 - val
_accuracy: 0.3661 - val_f1_m: 0.3651 - val_precision_m: 0.3666 - val_recall_m: 0.3643
Epoch 6/50
2856/2856 [==============================] - 492s 172ms/step - loss: 0.8072 - accuracy: 0.7132 - f1_m: 0.6531 - precision_m: 0.7180 - recall_m: 0.6206 - val_loss: 10.2742 - va
l_accuracy: 0.2517 - val_f1_m: 0.2508 - val_precision_m: 0.2508 - val_recall_m: 0.2508
Epoch 7/50
```

In [68]:
```python
dl_acc = hist2a.history["val_accuracy"][49]
dl_prec = hist2a.history["val_precision_m"][49]
dl_rec = hist2a.history["val_recall_m"][49]
dl_f1 = hist2a.history["val_f1_m"][49]

storeResults('VisionTransformer - EANet',dl_acc,dl_prec,dl_rec,dl_f1)
```

In [46]:
```
x=hist2a
plt.figure(figsize=(20,10))
plt.subplot(1, 2, 1)
plt.suptitle('Optimizer : adam', fontsize=10)
plt.ylabel('Loss', fontsize=16)
plt.plot(x.history['loss'], label='Training Loss')
plt.plot(x.history['val_loss'], label='Validation Loss')
plt.legend(loc='upper right')

plt.subplot(1, 2, 2)
plt.ylabel('Accuracy', fontsize=16)
plt.plot(x.history['accuracy'], label='Training Accuracy')
plt.plot(x.history['val_accuracy'], label='Validation Accuracy')
plt.legend(loc='lower right')
plt.show()
```



TL Models

In [69]:
```
from tensorflow.keras.layers import Dense, Flatten, Input, Lambda
from tensorflow.keras.models import Model ,Sequential
from tensorflow.keras.preprocessing import image
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Dense, GlobalAveragePooling2D
from tensorflow.keras.applications import VGG16
from tensorflow.keras.preprocessing.image import ImageDataGenerator, load_img
from glob import glob
import matplotlib.pyplot as plt
import numpy as np
import tensorflow as tf
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score
```

In [24]:
```
from tensorflow.keras.layers import *
from tensorflow.keras.losses import *
from tensorflow.keras.models import *
from tensorflow.keras.optimizers import *
from sklearn.metrics import confusion_matrix
```

In [25]:
```
# Resizinig all the images to (224,224)
IMAGE_SIZE = [256,256]

train_path = 'Brain_Tumor_MRI_Image_Dataset//Training'
test_path = 'Brain_Tumor_MRI_Image_Dataset//Testing'
```

In [26]:
```
# Scaling all the images between 0 to 1

train_datagen = ImageDataGenerator(rescale = 1./255, shear_range=0.2, zoom_range=0.2, horizontal_flip=False)

# Performing only scaling on the test dataset

test_datagen = ImageDataGenerator(rescale=1./255)
```

In [27]:
```
train_set = train_datagen.flow_from_directory(train_path,
                                              target_size=(256,256),
                                              batch_size=2,
                                              class_mode = 'categorical')

test_set = test_datagen.flow_from_directory(test_path,
                                            target_size=(256,256),
                                            batch_size=2,
                                            class_mode='categorical')
```

```
Found 5712 images belonging to 4 classes.
Found 1311 images belonging to 4 classes.
```

VGG16

In [28]:
```
base_model = VGG16(input_shape = (256,256,3), weights=None, include_top=True)

x1= Flatten()(base_model.output)
prediction1 = Dense(4, activation='softmax')(x1)

model3 = Model(inputs = base_model.inputs, outputs = prediction1)
model3.summary()
model3.compile(loss = 'categorical_crossentropy', optimizer=Adam(learning_rate=0.0001), metrics=["accuracy",f1_m,precision_m, recall_m])
```

```
Model: "model_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 input_3 (InputLayer)        [(None, 256, 256, 3)]     0         
                                                                 
 block1_conv1 (Conv2D)       (None, 256, 256, 64)      1792      
                                                                 
 block1_conv2 (Conv2D)       (None, 256, 256, 64)      36928     
                                                                 
 block1_pool (MaxPooling2D)  (None, 128, 128, 64)      0         
                                                                 
 block2_conv1 (Conv2D)       (None, 128, 128, 128)     73856     
                                                                 
 block2_conv2 (Conv2D)       (None, 128, 128, 128)     147584    
                                                                 
 block2_pool (MaxPooling2D)  (None, 64, 64, 128)       0         
                                                                 
 block3_conv1 (Conv2D)       (None, 64, 64, 256)       295168    
                                                                 
 block3_conv2 (Conv2D)       (None, 64, 64, 256)       590080    
                                                                 
 block3_conv3 (Conv2D)       (None, 64, 64, 256)       590080    
                                                                 
 block3_pool (MaxPooling2D)  (None, 32, 32, 256)       0         
                                                                 
 block4_conv1 (Conv2D)       (None, 32, 32, 512)       1180160   
                                                                 
 block4_conv2 (Conv2D)       (None, 32, 32, 512)       2359808   
                                                                 
 block4_conv3 (Conv2D)       (None, 32, 32, 512)       2359808   
                                                                 
 block4_pool (MaxPooling2D)  (None, 16, 16, 512)       0         
                                                                 
 block5_conv1 (Conv2D)       (None, 16, 16, 512)       2359808   
                                                                 
 block5_conv2 (Conv2D)       (None, 16, 16, 512)       2359808   
                                                                 
 block5_conv3 (Conv2D)       (None, 16, 16, 512)       2359808   
                                                                 
 block5_pool (MaxPooling2D)  (None, 8, 8, 512)         0         
                                                                 
 flatten (Flatten)           (None, 32768)             0         
                                                                 
 fc1 (Dense)                 (None, 4096)              134221824 
                                                                 
 fc2 (Dense)                 (None, 4096)              16781312  
                                                                 
 predictions (Dense)         (None, 1000)              4097000   
                                                                 
 flatten_2 (Flatten)         (None, 1000)              0         
                                                                 
 dense_2 (Dense)             (None, 4)                 4004      
                                                                 
=================================================================
Total params: 169,818,828
Trainable params: 169,818,828
Non-trainable params: 0
_________________________________________________________________
```

In [29]:
```
hist3 = model3.fit(train_set, epochs=50, validation_data=test_set,steps_per_epoch=len(train_set), validation_steps=len(test_set),callbacks=[learning_rate_reduction, early_stop])
```

```
Epoch 1/50
2856/2856 [==============================] - 82s 28ms/step - loss: 1.3843 - accuracy: 0.2771 - f1_m: 0.0000e+00 - precision_m: 0.0000e+00 - recall_m: 0.0000e+00 - val_loss: 1.38
21 - val_accuracy: 0.3089 - val_f1_m: 0.0000e+00 - val_precision_m: 0.0000e+00 - val_recall_m: 0.0000e+00 - lr: 1.0000e-04
Epoch 2/50
2856/2856 [==============================] - 79s 28ms/step - loss: 1.3837 - accuracy: 0.2792 - f1_m: 0.0000e+00 - precision_m: 0.0000e+00 - recall_m: 0.0000e+00 - val_loss: 1.38
14 - val_accuracy: 0.3089 - val_f1_m: 0.0000e+00 - val_precision_m: 0.0000e+00 - val_recall_m: 0.0000e+00 - lr: 1.0000e-04
Epoch 3/50
2856/2856 [==============================] - 79s 28ms/step - loss: 1.3836 - accuracy: 0.2792 - f1_m: 0.0000e+00 - precision_m: 0.0000e+00 - recall_m: 0.0000e+00 - val_loss: 1.38
10 - val_accuracy: 0.3089 - val_f1_m: 0.0000e+00 - val_precision_m: 0.0000e+00 - val_recall_m: 0.0000e+00 - lr: 1.0000e-04
Epoch 4/50
2856/2856 [==============================] - ETA: 0s - loss: 1.3835 - accuracy: 0.2792 - f1_m: 0.0000e+00 - precision_m: 0.0000e+00 - recall_m: 0.0000e+00
Epoch 4: ReduceLROnPlateau reducing learning rate to 2.9999999242136255e-05.
2856/2856 [==============================] - 83s 29ms/step - loss: 1.3835 - accuracy: 0.2792 - f1_m: 0.0000e+00 - precision_m: 0.0000e+00 - recall_m: 0.0000e+00 - val_loss: 1.38
08 - val_accuracy: 0.3089 - val_f1_m: 0.0000e+00 - val_precision_m: 0.0000e+00 - val_recall_m: 0.0000e+00 - lr: 1.0000e-04
Epoch 5/50
2856/2856 [==============================] - 83s 29ms/step - loss: 1.3834 - accuracy: 0.2792 - f1_m: 0.0000e+00 - precision_m: 0.0000e+00 - recall_m: 0.0000e+00 - val_loss: 1.38
08 - val_accuracy: 0.3089 - val_f1_m: 0.0000e+00 - val_precision_m: 0.0000e+00 - val_recall_m: 0.0000e+00 - lr: 3.0000e-05
Epoch 6/50
2856/2856 [==============================] - 84s 29ms/step - loss: 1.3834 - accuracy: 0.2792 - f1_m: 0.0000e+00 - precision_m: 0.0000e+00 - recall_m: 0.0000e+00 - val_loss: 1.38
07 - val_accuracy: 0.3089 - val_f1_m: 0.0000e+00 - val_precision_m: 0.0000e+00 - val_recall_m: 0.0000e+00 - lr: 3.0000e-05
Epoch 7/50
2855/2856 [============================>.] - ETA: 0s - loss: 1.3834 - accuracy: 0.2792 - f1_m: 0.0000e+00 - precision_m: 0.0000e+00 - recall_m: 0.0000e+00
Epoch 7: ReduceLROnPlateau reducing learning rate to 8.999999772640877e-06.
2856/2856 [==============================] - 86s 30ms/step - loss: 1.3834 - accuracy: 0.2792 - f1_m: 0.0000e+00 - precision_m: 0.0000e+00 - recall_m: 0.0000e+00 - val_loss: 1.38
07 - val_accuracy: 0.3089 - val_f1_m: 0.0000e+00 - val_precision_m: 0.0000e+00 - val_recall_m: 0.0000e+00 - lr: 3.0000e-05
Epoch 8/50
2856/2856 [==============================] - 83s 29ms/step - loss: 1.3834 - accuracy: 0.2792 - f1_m: 0.0000e+00 - precision_m: 0.0000e+00 - recall_m: 0.0000e+00 - val_loss: 1.38
07 - val_accuracy: 0.3089 - val_f1_m: 0.0000e+00 - val_precision_m: 0.0000e+00 - val_recall_m: 0.0000e+00 - lr: 9.0000e-06
Epoch 9/50
2856/2856 [==============================] - 83s 29ms/step - loss: 1.3834 - accuracy: 0.2792 - f1_m: 0.0000e+00 - precision_m: 0.0000e+00 - recall_m: 0.0000e+00 - val_loss: 1.38
06 - val_accuracy: 0.3089 - val_f1_m: 0.0000e+00 - val_precision_m: 0.0000e+00 - val_recall_m: 0.0000e+00 - lr: 9.0000e-06
Epoch 10/50
2856/2856 [==============================] - ETA: 0s - loss: 1.3834 - accuracy: 0.2792 - f1_m: 0.0000e+00 - precision_m: 0.0000e+00 - recall_m: 0.0000e+00
Epoch 10: ReduceLROnPlateau reducing learning rate to 2.6999998226528985e-06.
2856/2856 [==============================] - 83s 29ms/step - loss: 1.3834 - accuracy: 0.2792 - f1_m: 0.0000e+00 - precision_m: 0.0000e+00 - recall_m: 0.0000e+00 - val_loss: 1.38
06 - val_accuracy: 0.3089 - val_f1_m: 0.0000e+00 - val_precision_m: 0.0000e+00 - val_recall_m: 0.0000e+00 - lr: 9.0000e-06
Epoch 11/50
2855/2856 [============================>.] - ETA: 0s - loss: 1.3834 - accuracy: 0.2793 - f1_m: 0.0000e+00 - precision_m: 0.0000e+00 - recall_m: 0.0000e+00Restoring model weights
from the end of the best epoch: 1.
2856/2856 [==============================] - 84s 29ms/step - loss: 1.3834 - accuracy: 0.2792 - f1_m: 0.0000e+00 - precision_m: 0.0000e+00 - recall_m: 0.0000e+00 - val_loss: 1.38
06 - val_accuracy: 0.3089 - val_f1_m: 0.0000e+00 - val_precision_m: 0.0000e+00 - val_recall_m: 0.0000e+00 - lr: 2.7000e-06
Epoch 11: early stopping
```

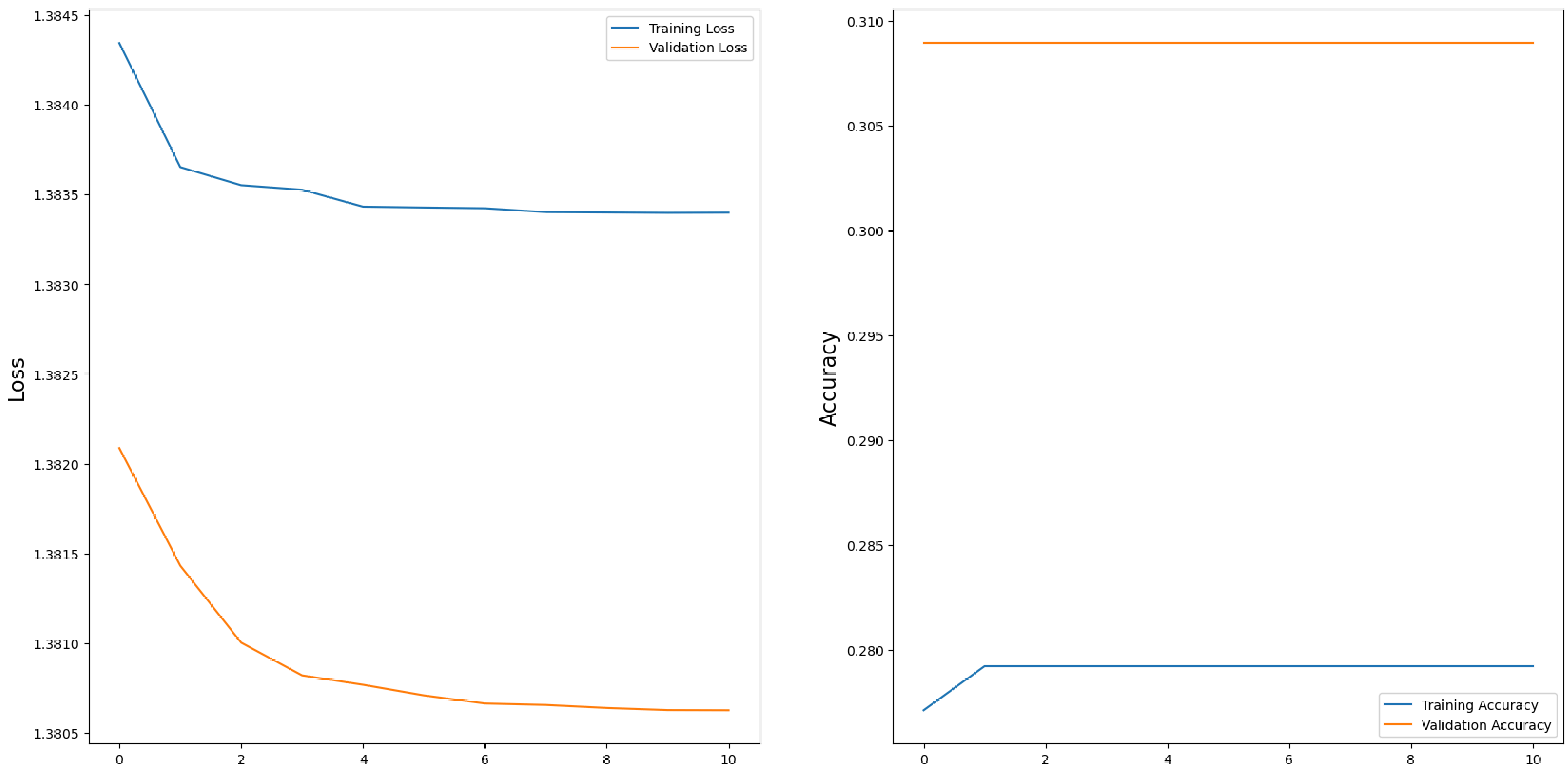
In [70]:
```
predictions = model3.predict(test_set)
y_pred = np.argmax(predictions, axis=1)
y_true = test_set.classes

dl_acc = hist3.history["val_accuracy"][10]
dl_prec = precision_score(y_true, y_pred, average='weighted')
dl_rec = recall_score(y_true, y_pred, average='weighted')
dl_f1 = f1_score(y_true, y_pred, average='weighted')

storeResults('TL - VGG16',dl_acc,dl_prec,dl_rec,dl_f1)
```

```
656/656 [==============================] - 4s 6ms/step
```

In [30]:
```
model3.save('models/vgg16.h5')
```

In [31]:
```
x=hist3
plt.figure(figsize=(20,10))
plt.subplot(1, 2, 1)
plt.suptitle('Optimizer : adam', fontsize=10)
plt.ylabel('Loss', fontsize=16)
plt.plot(x.history['loss'], label='Training Loss')
plt.plot(x.history['val_loss'], label='Validation Loss')
plt.legend(loc='upper right')

plt.subplot(1, 2, 2)
plt.ylabel('Accuracy', fontsize=16)
plt.plot(x.history['accuracy'], label='Training Accuracy')
plt.plot(x.history['val_accuracy'], label='Validation Accuracy')
plt.legend(loc='lower right')
plt.show()
```

Optimizer : adam



VGG19

In [32]:
```
base_model = VGG19(input_shape = (256,256,3), weights=None, include_top=True)

x1= Flatten()(base_model.output)
prediction1 = Dense(4, activation='softmax')(x1)

model4 = Model(inputs = base_model.inputs, outputs = prediction1)
model4.summary()
model4.compile(loss = 'categorical_crossentropy', optimizer='sgd', metrics=["accuracy",f1_m,precision_m, recall_m])
```

```
Model: "model_3"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 input_4 (InputLayer)        [(None, 256, 256, 3)]     0         
                                                                 
 block1_conv1 (Conv2D)       (None, 256, 256, 64)      1792      
                                                                 
 block1_conv2 (Conv2D)       (None, 256, 256, 64)      36928     
                                                                 
 block1_pool (MaxPooling2D)  (None, 128, 128, 64)      0         
                                                                 
 block2_conv1 (Conv2D)       (None, 128, 128, 128)     73856     
                                                                 
 block2_conv2 (Conv2D)       (None, 128, 128, 128)     147584    
                                                                 
 block2_pool (MaxPooling2D)  (None, 64, 64, 128)       0         
                                                                 
 block3_conv1 (Conv2D)       (None, 64, 64, 256)       295168    
                                                                 
 block3_conv2 (Conv2D)       (None, 64, 64, 256)       590080    
                                                                 
 block3_conv3 (Conv2D)       (None, 64, 64, 256)       590080    
                                                                 
 block3_conv4 (Conv2D)       (None, 64, 64, 256)       590080    
                                                                 
 block3_pool (MaxPooling2D)  (None, 32, 32, 256)       0         
                                                                 
 block4_conv1 (Conv2D)       (None, 32, 32, 512)       1180160   
                                                                 
 block4_conv2 (Conv2D)       (None, 32, 32, 512)       2359808   
                                                                 
 block4_conv3 (Conv2D)       (None, 32, 32, 512)       2359808   
                                                                 
 block4_conv4 (Conv2D)       (None, 32, 32, 512)       2359808   
                                                                 
 block4_pool (MaxPooling2D)  (None, 16, 16, 512)       0         
                                                                 
 block5_conv1 (Conv2D)       (None, 16, 16, 512)       2359808   
                                                                 
 block5_conv2 (Conv2D)       (None, 16, 16, 512)       2359808   
                                                                 
 block5_conv3 (Conv2D)       (None, 16, 16, 512)       2359808   
                                                                 
 block5_conv4 (Conv2D)       (None, 16, 16, 512)       2359808   
                                                                 
 block5_pool (MaxPooling2D)  (None, 8, 8, 512)         0         
                                                                 
 flatten (Flatten)           (None, 32768)             0         
                                                                 
 fc1 (Dense)                 (None, 4096)              134221824 
                                                                 
 fc2 (Dense)                 (None, 4096)              16781312  
                                                                 
 predictions (Dense)         (None, 1000)              4097000   
                                                                 
 flatten_3 (Flatten)         (None, 1000)              0         
                                                                 
 dense_3 (Dense)             (None, 4)                 4004      
                                                                 
=================================================================
Total params: 175,128,524
Trainable params: 175,128,524
Non-trainable params: 0
_________________________________________________________________
```

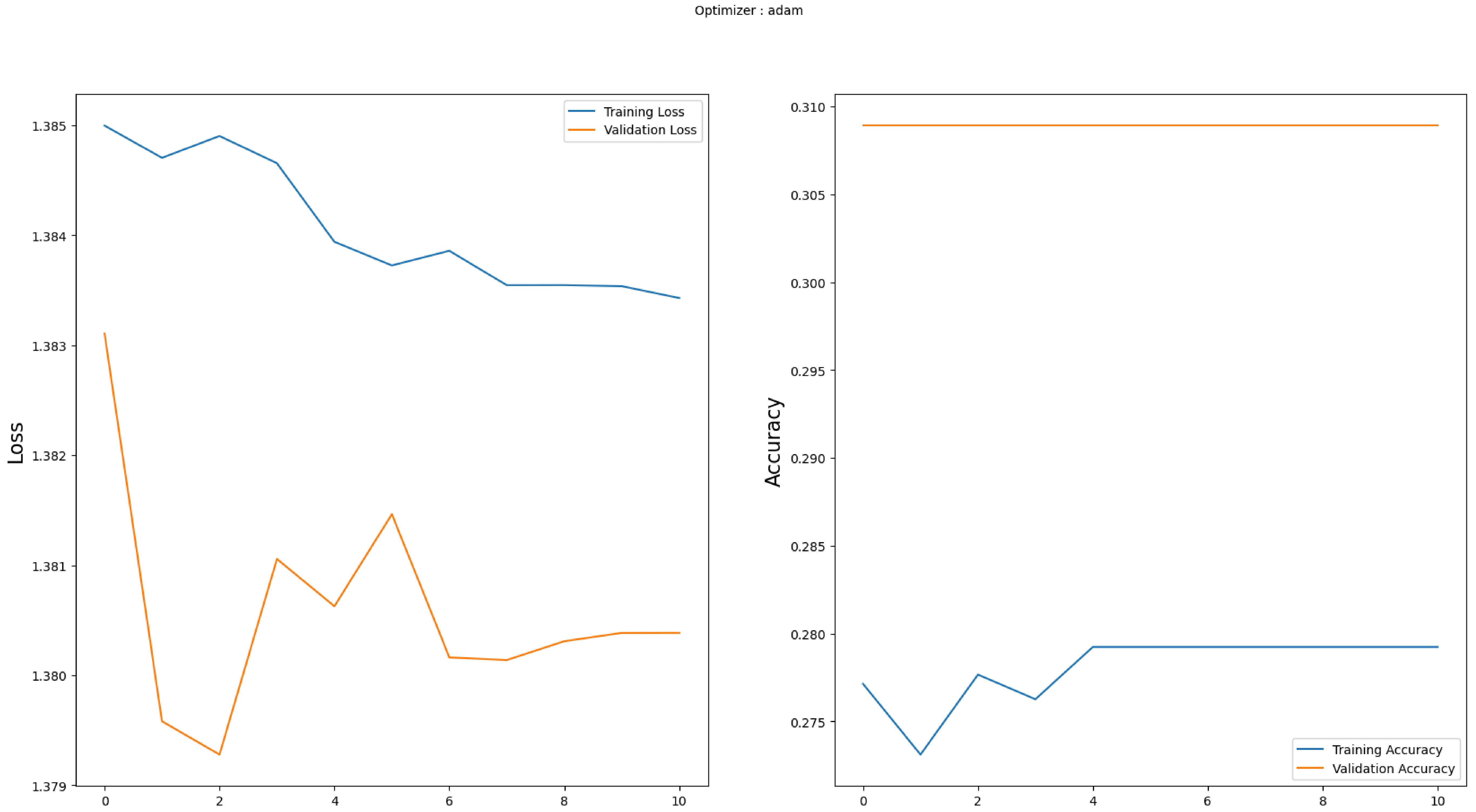
In [33]:
```
hist4 = model4.fit(train_set, epochs=50, validation_data=test_set,steps_per_epoch=len(train_set), validation_steps=len(test_set),callbacks=[learning_rate_reduction, early_stop])
```

```
Epoch 1/50
2856/2856 [==============================] - 79s 27ms/step - loss: 1.3850 - accuracy: 0.2771 - f1_m: 0.0000e+00 - precision_m: 0.0000e+00 - recall_m: 0.0000e+00 - val_loss: 1.38
31 - val_accuracy: 0.3089 - val_f1_m: 0.0000e+00 - val_precision_m: 0.0000e+00 - val_recall_m: 0.0000e+00 - lr: 0.0100
Epoch 2/50
2856/2856 [==============================] - 77s 27ms/step - loss: 1.3847 - accuracy: 0.2731 - f1_m: 0.0000e+00 - precision_m: 0.0000e+00 - recall_m: 0.0000e+00 - val_loss: 1.37
96 - val_accuracy: 0.3089 - val_f1_m: 0.0000e+00 - val_precision_m: 0.0000e+00 - val_recall_m: 0.0000e+00 - lr: 0.0100
Epoch 3/50
2856/2856 [==============================] - 77s 27ms/step - loss: 1.3849 - accuracy: 0.2777 - f1_m: 0.0000e+00 - precision_m: 0.0000e+00 - recall_m: 0.0000e+00 - val_loss: 1.37
93 - val_accuracy: 0.3089 - val_f1_m: 0.0000e+00 - val_precision_m: 0.0000e+00 - val_recall_m: 0.0000e+00 - lr: 0.0100
Epoch 4/50
2855/2856 [============================>.] - ETA: 0s - loss: 1.3847 - accuracy: 0.2764 - f1_m: 0.0000e+00 - precision_m: 0.0000e+00 - recall_m: 0.0000e+00
Epoch 4: ReduceLROnPlateau reducing learning rate to 0.0029999999329447745.
2856/2856 [==============================] - 77s 27ms/step - loss: 1.3847 - accuracy: 0.2763 - f1_m: 0.0000e+00 - precision_m: 0.0000e+00 - recall_m: 0.0000e+00 - val_loss: 1.38
11 - val_accuracy: 0.3089 - val_f1_m: 0.0000e+00 - val_precision_m: 0.0000e+00 - val_recall_m: 0.0000e+00 - lr: 0.0100
Epoch 5/50
2856/2856 [==============================] - 77s 27ms/step - loss: 1.3839 - accuracy: 0.2792 - f1_m: 0.0000e+00 - precision_m: 0.0000e+00 - recall_m: 0.0000e+00 - val_loss: 1.38
06 - val_accuracy: 0.3089 - val_f1_m: 0.0000e+00 - val_precision_m: 0.0000e+00 - val_recall_m: 0.0000e+00 - lr: 0.0030
Epoch 6/50
2856/2856 [==============================] - 77s 27ms/step - loss: 1.3837 - accuracy: 0.2792 - f1_m: 0.0000e+00 - precision_m: 0.0000e+00 - recall_m: 0.0000e+00 - val_loss: 1.38
15 - val_accuracy: 0.3089 - val_f1_m: 0.0000e+00 - val_precision_m: 0.0000e+00 - val_recall_m: 0.0000e+00 - lr: 0.0030
Epoch 7/50
2855/2856 [============================>.] - ETA: 0s - loss: 1.3839 - accuracy: 0.2793 - f1_m: 0.0000e+00 - precision_m: 0.0000e+00 - recall_m: 0.0000e+00
Epoch 7: ReduceLROnPlateau reducing learning rate to 0.0009000000078231095.
2856/2856 [==============================] - 77s 27ms/step - loss: 1.3839 - accuracy: 0.2792 - f1_m: 0.0000e+00 - precision_m: 0.0000e+00 - recall_m: 0.0000e+00 - val_loss: 1.38
02 - val_accuracy: 0.3089 - val_f1_m: 0.0000e+00 - val_precision_m: 0.0000e+00 - val_recall_m: 0.0000e+00 - lr: 0.0030
Epoch 8/50
2856/2856 [==============================] - 78s 27ms/step - loss: 1.3835 - accuracy: 0.2792 - f1_m: 0.0000e+00 - precision_m: 0.0000e+00 - recall_m: 0.0000e+00 - val_loss: 1.38
01 - val_accuracy: 0.3089 - val_f1_m: 0.0000e+00 - val_precision_m: 0.0000e+00 - val_recall_m: 0.0000e+00 - lr: 9.0000e-04
Epoch 9/50
2856/2856 [==============================] - 77s 27ms/step - loss: 1.3835 - accuracy: 0.2792 - f1_m: 0.0000e+00 - precision_m: 0.0000e+00 - recall_m: 0.0000e+00 - val_loss: 1.38
03 - val_accuracy: 0.3089 - val_f1_m: 0.0000e+00 - val_precision_m: 0.0000e+00 - val_recall_m: 0.0000e+00 - lr: 9.0000e-04
Epoch 10/50
2855/2856 [============================>.] - ETA: 0s - loss: 1.3835 - accuracy: 0.2793 - f1_m: 0.0000e+00 - precision_m: 0.0000e+00 - recall_m: 0.0000e+00
Epoch 10: ReduceLROnPlateau reducing learning rate to 0.00026999999536201356.
2856/2856 [==============================] - 77s 27ms/step - loss: 1.3835 - accuracy: 0.2792 - f1_m: 0.0000e+00 - precision_m: 0.0000e+00 - recall_m: 0.0000e+00 - val_loss: 1.38
04 - val_accuracy: 0.3089 - val_f1_m: 0.0000e+00 - val_precision_m: 0.0000e+00 - val_recall_m: 0.0000e+00 - lr: 9.0000e-04
Epoch 11/50
2856/2856 [==============================] - ETA: 0s - loss: 1.3834 - accuracy: 0.2792 - f1_m: 0.0000e+00 - precision_m: 0.0000e+00 - recall_m: 0.0000e+00Restoring model weights
from the end of the best epoch: 1.
2856/2856 [==============================] - 77s 27ms/step - loss: 1.3834 - accuracy: 0.2792 - f1_m: 0.0000e+00 - precision_m: 0.0000e+00 - recall_m: 0.0000e+00 - val_loss: 1.38
04 - val_accuracy: 0.3089 - val_f1_m: 0.0000e+00 - val_precision_m: 0.0000e+00 - val_recall_m: 0.0000e+00 - lr: 2.7000e-04
Epoch 11: early stopping
```

In [71]:
```
predictions = model4.predict(test_set)
y_pred = np.argmax(predictions, axis=1)
y_true = test_set.classes

dl_acc = hist4.history["val_accuracy"][10]
dl_prec = precision_score(y_true, y_pred, average='weighted')
dl_rec = recall_score(y_true, y_pred, average='weighted')
dl_f1 = f1_score(y_true, y_pred, average='weighted')

storeResults('TL - VGG19',dl_acc,dl_prec,dl_rec,dl_f1)
```

```
656/656 [==============================] - 4s 6ms/step
```

In [34]:
```
x=hist4
plt.figure(figsize=(20,10))
plt.subplot(1, 2, 1)
plt.suptitle('Optimizer : adam', fontsize=10)
plt.ylabel('Loss', fontsize=16)
plt.plot(x.history['loss'], label='Training Loss')
plt.plot(x.history['val_loss'], label='Validation Loss')
plt.legend(loc='upper right')

plt.subplot(1, 2, 2)
plt.ylabel('Accuracy', fontsize=16)
plt.plot(x.history['accuracy'], label='Training Accuracy')
plt.plot(x.history['val_accuracy'], label='Validation Accuracy')
plt.legend(loc='lower right')
plt.show()
```



InceptionV3

In [35]:
```
base_model = InceptionV3(input_shape = (256,256,3), weights=None, include_top=True)

x1= Flatten()(base_model.output)
prediction1 = Dense(4, activation='softmax')(x1)

model5 = Model(inputs = base_model.inputs, outputs = prediction1)
model5.summary()
model5.compile(loss = 'categorical_crossentropy', optimizer='sgd', metrics=["accuracy",f1_m,precision_m, recall_m])
```

```
Model: "model_4"
__________________________________________________________________________________________________
 Layer (type)                   Output Shape         Param #     Connected to
==================================================================================================
 input_5 (InputLayer)           [(None, 256, 256, 3  0           []
                                )]

 conv2d (Conv2D)                (None, 127, 127, 32  864         ['input_5[0][0]']
                                )

 batch_normalization (BatchNorm  (None, 127, 127, 32  96          ['conv2d[0][0]']
 alization)                     )

 activation (Activation)        (None, 127, 127, 32  0           ['batch_normalization[0][0]']
                                )

 conv2d_1 (Conv2D)              (None, 125, 125, 32  9216        ['activation[0][0]']
                                )
```

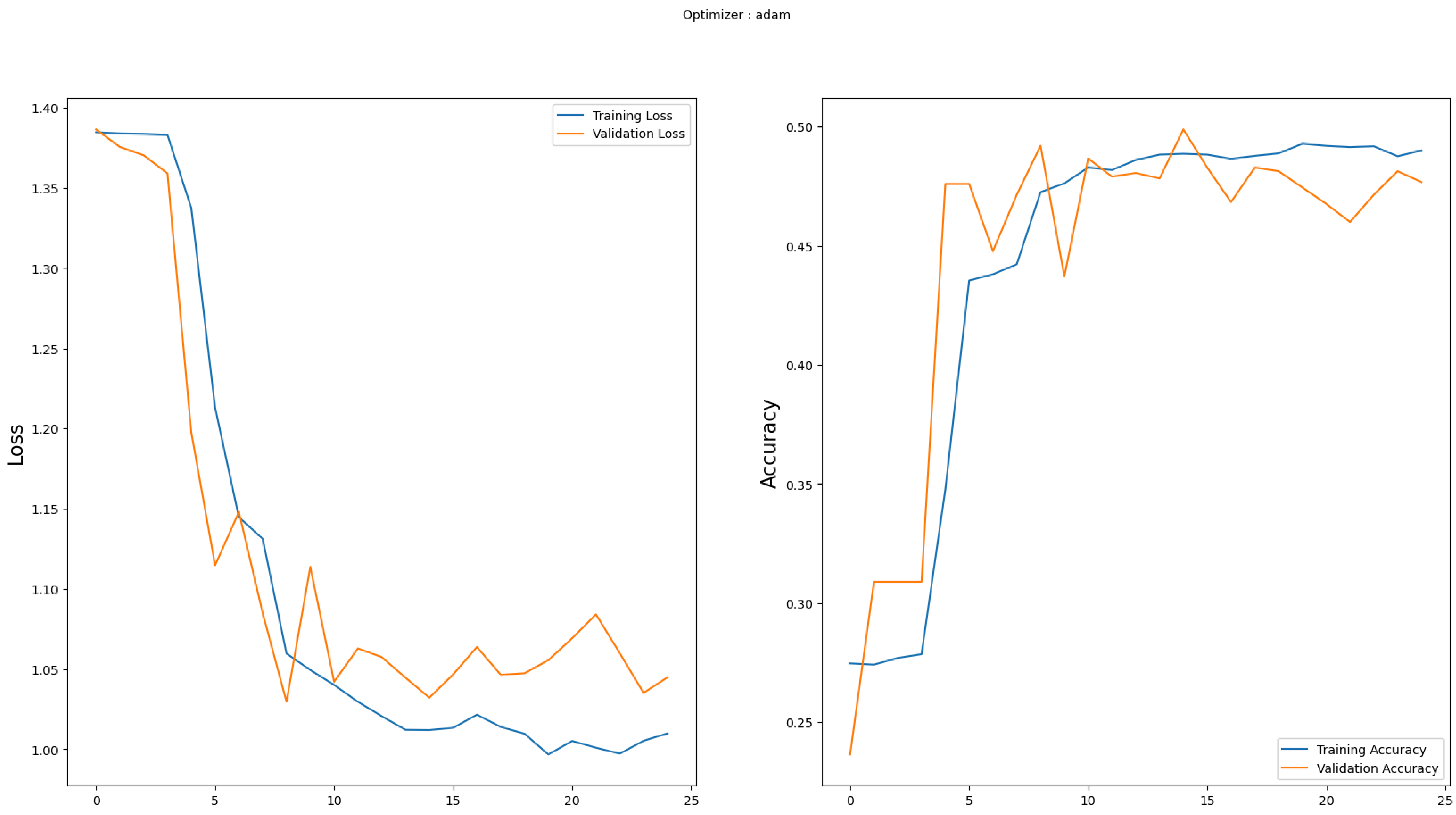
In [36]:
```
hist5 = model5.fit(train_set, epochs=50, validation_data=test_set,steps_per_epoch=len(train_set), validation_steps=len(test_set),callbacks=[learning_rate_reduction, early_stop])
```

```
Epoch 1/50
2856/2856 [==============================] - 160s 54ms/step - loss: 1.3848 - accuracy: 0.2747 - f1_m: 0.0000e+00 - precision_m: 0.0000e+00 - recall_m: 0.0000e+00 - val_loss: 1.3866 - val_accuracy: 0.2365 - val_f1_m: 0.0000e+00 - val_precision_m: 0.0000e+00 - val_recall_m: 0.0000e+00 - lr: 0.0100
Epoch 2/50
2856/2856 [==============================] - 153s 53ms/step - loss: 1.3841 - accuracy: 0.2742 - f1_m: 0.0000e+00 - precision_m: 0.0000e+00 - recall_m: 0.0000e+00 - val_loss: 1.3756 - val_accuracy: 0.3089 - val_f1_m: 0.0000e+00 - val_precision_m: 0.0000e+00 - val_recall_m: 0.0000e+00 - lr: 0.0100
Epoch 3/50
2856/2856 [==============================] - 153s 53ms/step - loss: 1.3838 - accuracy: 0.2770 - f1_m: 0.0000e+00 - precision_m: 0.0000e+00 - recall_m: 0.0000e+00 - val_loss: 1.3704 - val_accuracy: 0.3089 - val_f1_m: 0.0000e+00 - val_precision_m: 0.0000e+00 - val_recall_m: 0.0000e+00 - lr: 0.0100
Epoch 4/50
2856/2856 [==============================] - 152s 53ms/step - loss: 1.3832 - accuracy: 0.2785 - f1_m: 0.0000e+00 - precision_m: 0.0000e+00 - recall_m: 0.0000e+00 - val_loss: 1.3591 - val_accuracy: 0.3089 - val_f1_m: 0.0000e+00 - val_precision_m: 0.0000e+00 - val_recall_m: 0.0000e+00 - lr: 0.0100
Epoch 5/50
2856/2856 [==============================] - 153s 54ms/step - loss: 1.3376 - accuracy: 0.3477 - f1_m: 0.0000e+00 - precision_m: 0.0000e+00 - recall_m: 0.0000e+00 - val_loss: 1.1976 - val_accuracy: 0.4760 - val_f1_m: 0.0000e+00 - val_precision_m: 0.0000e+00 - val_recall_m: 0.0000e+00 - lr: 0.0100
Epoch 6/50
2856/2856 [==============================] - 152s 53ms/step - loss: 1.2131 - accuracy: 0.4354 - f1_m: 0.1975 - precision_m: 0.2962 - recall_m: 0.1481 - val_loss: 1.1147 - val_accuracy: 0.4760 - val_f1_m: 0.3064 - val_precision_m: 0.4207 - val_recall_m: 0.2492 - lr: 0.0100
Epoch 7/50
```

In [72]:
```
dl_acc = hist5.history["val_accuracy"][24]
dl_prec = hist5.history["val_precision_m"][24]
dl_rec = hist5.history["val_recall_m"][24]
dl_f1 = hist5.history["val_f1_m"][24]

storeResults('TL - InceptionV3',dl_acc,dl_prec,dl_rec,dl_f1)
```

In [37]:
```
model5.save('models/inceptionv3.h5')
```

In [38]:
```
x=hist5
plt.figure(figsize=(20,10))
plt.subplot(1, 2, 1)
plt.suptitle('Optimizer : adam', fontsize=10)
plt.ylabel('Loss', fontsize=16)
plt.plot(x.history['loss'], label='Training Loss')
plt.plot(x.history['val_loss'], label='Validation Loss')
plt.legend(loc='upper right')

plt.subplot(1, 2, 2)
plt.ylabel('Accuracy', fontsize=16)
plt.plot(x.history['accuracy'], label='Training Accuracy')
plt.plot(x.history['val_accuracy'], label='Validation Accuracy')
plt.legend(loc='lower right')
plt.show()
```



ResNet50

In [39]:
```
base_model = ResNet50(input_shape = (256,256,3), weights=None, include_top=True)

x1= Flatten()(base_model.output)
prediction1 = Dense(4, activation='softmax')(x1)
model6 = Model(inputs = base_model.inputs, outputs = prediction1)
model6.summary()
model6.compile(loss = 'categorical_crossentropy', optimizer='sgd', metrics=["accuracy",f1_m,precision_m, recall_m])
```

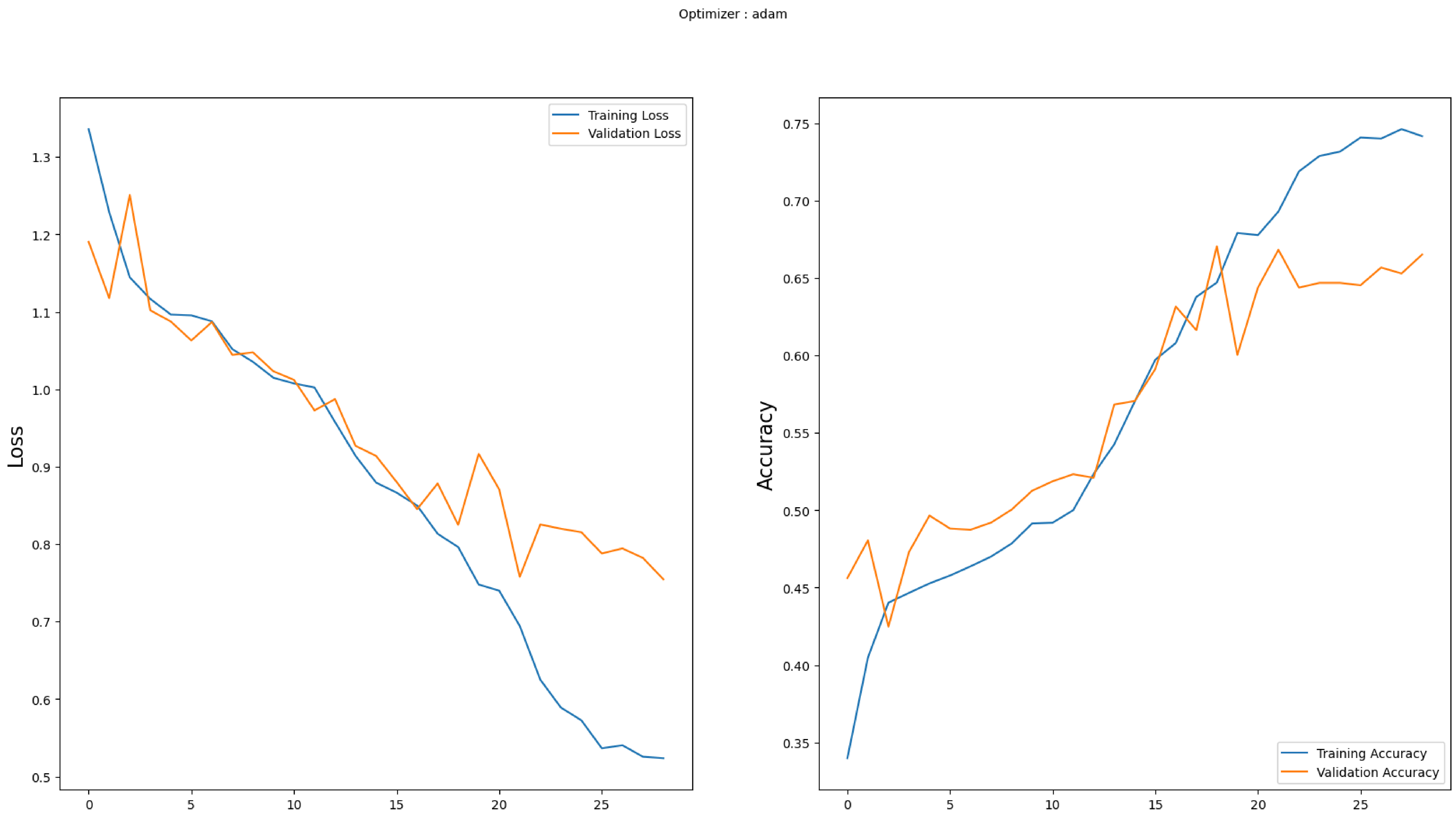
```
Model: "model_5"
__________________________________________________________________________________________________
 Layer (type)                   Output Shape         Param #     Connected to
==================================================================================================
 input_6 (InputLayer)           [(None, 256, 256, 3  0           []
                                )]

 conv1_pad (ZeroPadding2D)      (None, 262, 262, 3)  0           ['input_6[0][0]']

 conv1_conv (Conv2D)            (None, 128, 128, 64  9472        ['conv1_pad[0][0]']
                                )

 conv1_bn (BatchNormalization)  (None, 128, 128, 64  256         ['conv1_conv[0][0]']
                                )

 conv1_relu (Activation)        (None, 128, 128, 64  0           ['conv1_bn[0][0]']
                                )

 pool1_pad (ZeroPadding2D)      (None, 130, 130, 64  0           ['conv1_relu[0][0]']
```

In [40]:
```
hist6 = model6.fit(train_set, epochs=50, validation_data=test_set,steps_per_epoch=len(train_set), validation_steps=len(test_set),callbacks=[learning_rate_reduction, early_stop])
```

```
Epoch 1/50
2856/2856 [==============================] - 106s 35ms/step - loss: 1.3359 - accuracy: 0.3398 - f1_m: 0.0082 - precision_m: 0.0123 - recall_m: 0.0061 - val_loss: 1.1903 - val_accuracy: 0.4561 - val_f1_m: 0.2734 - val_precision_m: 0.3780 - val_recall_m: 0.2210 - lr: 0.0100
Epoch 2/50
2856/2856 [==============================] - 101s 35ms/step - loss: 1.2287 - accuracy: 0.4048 - f1_m: 0.2187 - precision_m: 0.3281 - recall_m: 0.1640 - val_loss: 1.1176 - val_accuracy: 0.4805 - val_f1_m: 0.3148 - val_precision_m: 0.4306 - val_recall_m: 0.2569 - lr: 0.0100
Epoch 3/50
2856/2856 [==============================] - 100s 35ms/step - loss: 1.1446 - accuracy: 0.4403 - f1_m: 0.2421 - precision_m: 0.3631 - recall_m: 0.1815 - val_loss: 1.2509 - val_accuracy: 0.4249 - val_f1_m: 0.3468 - val_precision_m: 0.4505 - val_recall_m: 0.2950 - lr: 0.0100
Epoch 4/50
2856/2856 [==============================] - 100s 35ms/step - loss: 1.1167 - accuracy: 0.4466 - f1_m: 0.2416 - precision_m: 0.3624 - recall_m: 0.1812 - val_loss: 1.1019 - val_accuracy: 0.4729 - val_f1_m: 0.2861 - val_precision_m: 0.3933 - val_recall_m: 0.2325 - lr: 0.0100
Epoch 5/50
2856/2856 [==============================] - 100s 35ms/step - loss: 1.0965 - accuracy: 0.4527 - f1_m: 0.2465 - precision_m: 0.3697 - recall_m: 0.1849 - val_loss: 1.0875 - val_accuracy: 0.4966 - val_f1_m: 0.3516 - val_precision_m: 0.4695 - val_recall_m: 0.2927 - lr: 0.0100
Epoch 6/50
2856/2856 [==============================] - 100s 35ms/step - loss: 1.0955 - accuracy: 0.4578 - f1_m: 0.2558 - precision_m: 0.3838 - recall_m: 0.1919 - val_loss: 1.0630 - val_accuracy: 0.4882 - val_f1_m: 0.3013 - val_precision_m: 0.4192 - val_recall_m: 0.2424 - lr: 0.0100
Epoch 7/50
```

In [73]:
```
dl_acc = hist6.history["val_accuracy"][28]
dl_prec = hist6.history["val_precision_m"][28]
dl_rec = hist6.history["val_recall_m"][28]
dl_f1 = hist6.history["val_f1_m"][28]

storeResults('TL - ResNet50',dl_acc,dl_prec,dl_rec,dl_f1)
```

In [41]:
```
x=hist6
plt.figure(figsize=(20,10))
plt.subplot(1, 2, 1)
plt.suptitle('Optimizer : adam', fontsize=10)
plt.ylabel('Loss', fontsize=16)
plt.plot(x.history['loss'], label='Training Loss')
plt.plot(x.history['val_loss'], label='Validation Loss')
plt.legend(loc='upper right')

plt.subplot(1, 2, 2)
plt.ylabel('Accuracy', fontsize=16)
plt.plot(x.history['accuracy'], label='Training Accuracy')
plt.plot(x.history['val_accuracy'], label='Validation Accuracy')
plt.legend(loc='lower right')
plt.show()
```



Inception ResNetV2

In [42]:
```
base_model = InceptionResNetV2(input_shape = (256,256,3), weights=None, include_top=True)

x1= Flatten()(base_model.output)
prediction1 = Dense(4, activation='softmax')(x1)
model7 = Model(inputs = base_model.inputs, outputs = prediction1)
model7.summary()
model7.compile(loss = 'categorical_crossentropy', optimizer='sgd', metrics=["accuracy",f1_m,precision_m, recall_m])
```

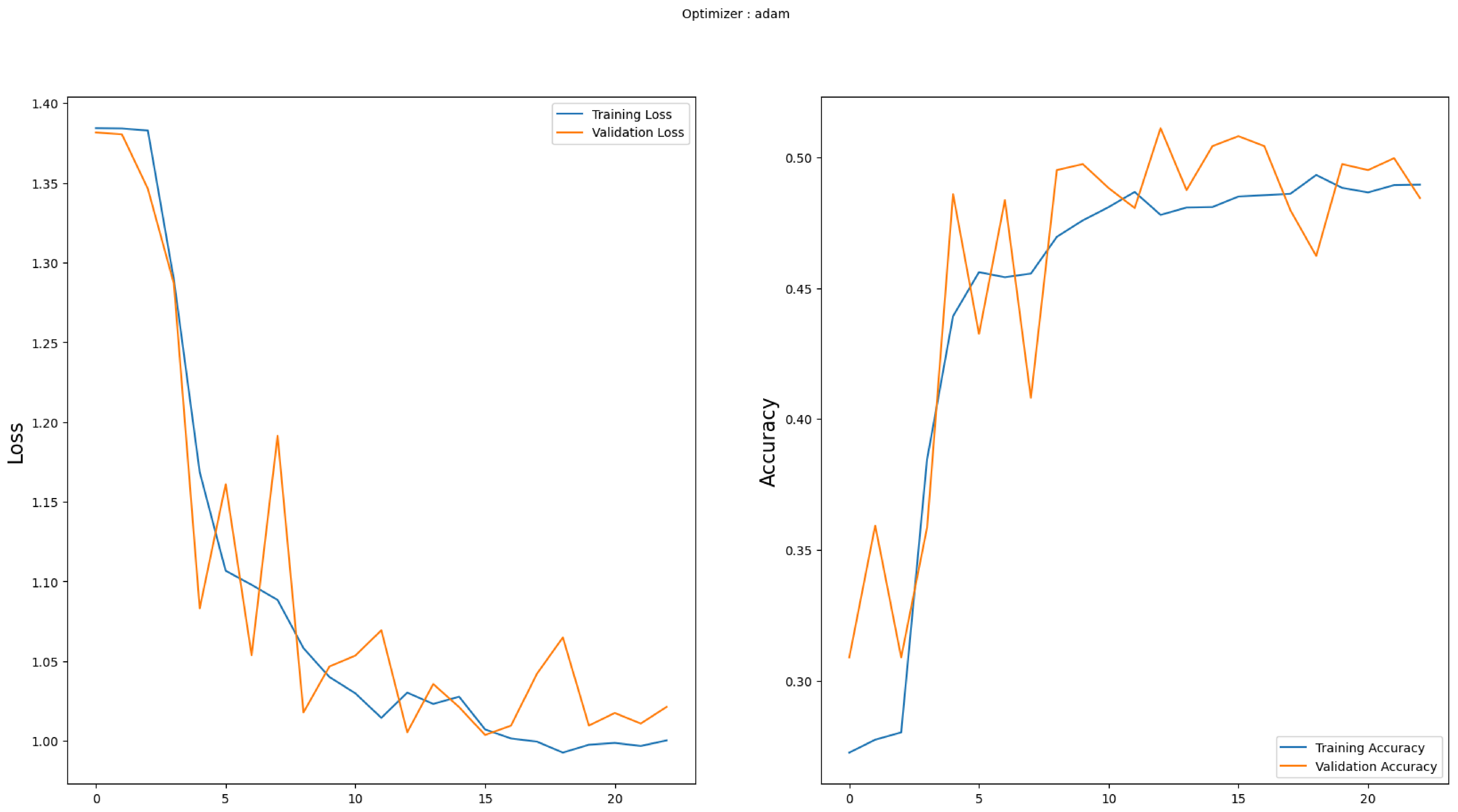
```
Model: "model_6"
__________________________________________________________________________________________________
 Layer (type)                   Output Shape         Param #     Connected to
==================================================================================================
 input_7 (InputLayer)           [(None, 256, 256, 3  0           []
                                )]

 conv2d_94 (Conv2D)             (None, 127, 127, 32  864         ['input_7[0][0]']
                                )

 batch_normalization_94 (BatchN  (None, 127, 127, 32  96         ['conv2d_94[0][0]']
 ormalization)                  )

 activation_94 (Activation)     (None, 127, 127, 32  0           ['batch_normalization_94[0][0]']
                                )

 conv2d_95 (Conv2D)             (None, 125, 125, 32  9216        ['activation_94[0][0]']
                                )
```

In [43]:
```
hist7 = model7.fit(train_set, epochs=50, validation_data=test_set,steps_per_epoch=len(train_set), validation_steps=len(test_set),callbacks=[learning_rate_reduction, early_stop])
```

```
Epoch 1/50
2856/2856 [==============================] - 341s 114ms/step - loss: 1.3843 - accuracy: 0.2726 - f1_m: 0.0000e+00 - precision_m: 0.0000e+00 - recall_m: 0.0000e+00 - val_loss: 1.3816 - val_accuracy: 0.3089 - val_f1_m: 0.0000e+00 - val_precision_m: 0.0000e+00 - val_recall_m: 0.0000e+00 - lr: 0.0100
Epoch 2/50
2856/2856 [==============================] - 323s 113ms/step - loss: 1.3841 - accuracy: 0.2775 - f1_m: 0.0000e+00 - precision_m: 0.0000e+00 - recall_m: 0.0000e+00 - val_loss: 1.3804 - val_accuracy: 0.3593 - val_f1_m: 0.0000e+00 - val_precision_m: 0.0000e+00 - val_recall_m: 0.0000e+00 - lr: 0.0100
Epoch 3/50
2856/2856 [==============================] - 323s 113ms/step - loss: 1.3829 - accuracy: 0.2803 - f1_m: 0.0000e+00 - precision_m: 0.0000e+00 - recall_m: 0.0000e+00 - val_loss: 1.3463 - val_accuracy: 0.3089 - val_f1_m: 0.0000e+00 - val_precision_m: 0.0000e+00 - val_recall_m: 0.0000e+00 - lr: 0.0100
Epoch 4/50
2856/2856 [==============================] - 323s 113ms/step - loss: 1.2896 - accuracy: 0.3846 - f1_m: 0.0770 - precision_m: 0.1155 - recall_m: 0.0578 - val_loss: 1.2871 - val_accuracy: 0.3585 - val_f1_m: 0.1118 - val_precision_m: 0.1646 - val_recall_m: 0.0854 - lr: 0.0100
Epoch 5/50
2856/2856 [==============================] - 324s 114ms/step - loss: 1.1685 - accuracy: 0.4393 - f1_m: 0.2465 - precision_m: 0.3697 - recall_m: 0.1849 - val_loss: 1.0830 - val_accuracy: 0.4859 - val_f1_m: 0.3039 - val_precision_m: 0.4146 - val_recall_m: 0.2485 - lr: 0.0100
Epoch 6/50
2856/2856 [==============================] - 323s 113ms/step - loss: 1.1066 - accuracy: 0.4561 - f1_m: 0.2722 - precision_m: 0.4083 - recall_m: 0.2041 - val_loss: 1.1609 - val_accuracy: 0.4325 - val_f1_m: 0.2401 - val_precision_m: 0.3316 - val_recall_m: 0.1944 - lr: 0.0100
Epoch 7/50
```

In [74]:
```
dl_acc = hist7.history["val_accuracy"][22]
dl_prec = hist7.history["val_precision_m"][22]
dl_rec = hist7.history["val_recall_m"][22]
dl_f1 = hist7.history["val_f1_m"][22]

storeResults('TL - InceptionResNetV2',dl_acc,dl_prec,dl_rec,dl_f1)
```

In [44]:
```
x=hist7
plt.figure(figsize=(20,10))
plt.subplot(1, 2, 1)
plt.suptitle('Optimizer : adam', fontsize=10)
plt.ylabel('Loss', fontsize=16)
plt.plot(x.history['loss'], label='Training Loss')
plt.plot(x.history['val_loss'], label='Validation Loss')
plt.legend(loc='upper right')

plt.subplot(1, 2, 2)
plt.ylabel('Accuracy', fontsize=16)
plt.plot(x.history['accuracy'], label='Training Accuracy')
plt.plot(x.history['val_accuracy'], label='Validation Accuracy')
plt.legend(loc='lower right')
plt.show()
```



Xception

In [45]:
```
# Defining the pretrained base model
base = Xception(include_top=False, weights='imagenet', input_shape=(256,256,3))
x = base.output
x = GlobalAveragePooling2D()(x)
# Defining the head of the model where the prediction is conducted
head = Dense(4, activation='softmax')(x)
# Combining base and head
model8 = Model(inputs=base.input, outputs=head)
model8.compile(optimizer='sgd',
              loss = 'categorical_crossentropy',
              metrics=["accuracy",f1_m,precision_m, recall_m])
model8.summary()
```

```
Model: "model_7"
__________________________________________________________________________________________________
 Layer (type)                   Output Shape         Param #     Connected to
==================================================================================================
 input_8 (InputLayer)           [(None, 256, 256, 3  0           []
                                )]

 block1_conv1 (Conv2D)          (None, 127, 127, 32  864         ['input_8[0][0]']
                                )

 block1_conv1_bn (BatchNormaliz  (None, 127, 127, 32  128        ['block1_conv1[0][0]']
 ation)                         )

 block1_conv1_act (Activation)  (None, 127, 127, 32  0           ['block1_conv1_bn[0][0]']
                                )

 block1_conv2 (Conv2D)          (None, 125, 125, 64  18432       ['block1_conv1_act[0][0]']
                                )
```

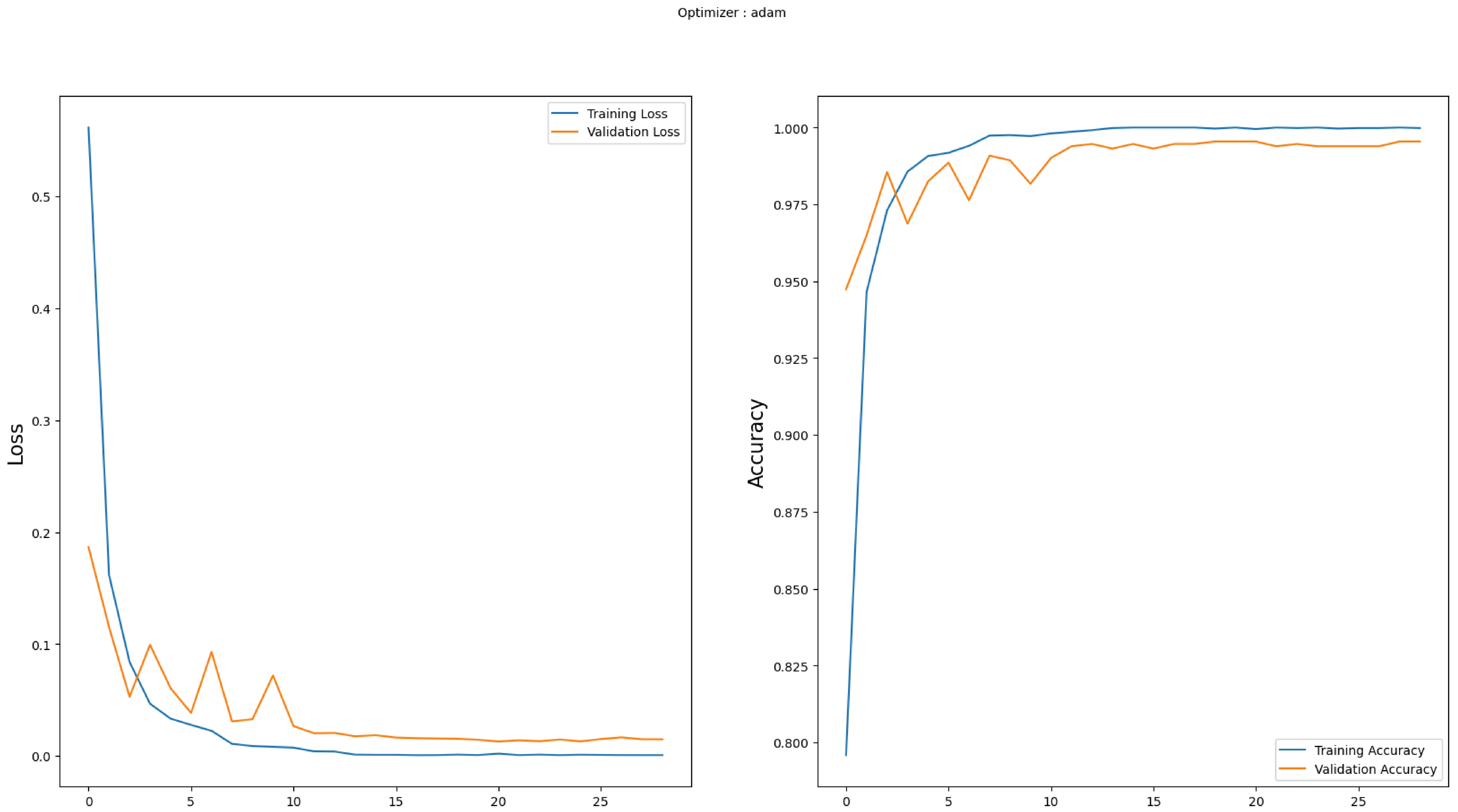
In [46]:
```
hist8 = model8.fit(train_set, epochs=50, validation_data=test_set,steps_per_epoch=len(train_set), validation_steps=len(test_set),callbacks=[learning_rate_reduction, early_stop])
```

```
Epoch 1/50
2856/2856 [==============================] - 86s 29ms/step - loss: 0.5615 - accuracy: 0.7959 - f1_m: 0.7437 - precision_m: 0.8046 - recall_m: 0.7132 - val_loss: 0.1866 - val_accuracy: 0.9474 - val_f1_m: 0.9418 - val_precision_m: 0.9520 - val_recall_m: 0.9367 - lr: 0.0100
Epoch 2/50
2856/2856 [==============================] - 81s 28ms/step - loss: 0.1618 - accuracy: 0.9464 - f1_m: 0.9442 - precision_m: 0.9561 - recall_m: 0.9382 - val_loss: 0.1152 - val_accuracy: 0.9649 - val_f1_m: 0.9652 - val_precision_m: 0.9688 - val_recall_m: 0.9634 - lr: 0.0100
Epoch 3/50
2856/2856 [==============================] - 81s 28ms/step - loss: 0.0840 - accuracy: 0.9730 - f1_m: 0.9716 - precision_m: 0.9767 - recall_m: 0.9690 - val_loss: 0.0529 - val_accuracy: 0.9855 - val_f1_m: 0.9860 - val_precision_m: 0.9870 - val_recall_m: 0.9855 - lr: 0.0100
Epoch 4/50
2856/2856 [==============================] - 80s 28ms/step - loss: 0.0467 - accuracy: 0.9856 - f1_m: 0.9854 - precision_m: 0.9862 - recall_m: 0.9849 - val_loss: 0.0993 - val_accuracy: 0.9687 - val_f1_m: 0.9688 - val_precision_m: 0.9688 - val_recall_m: 0.9688 - lr: 0.0100
Epoch 5/50
2856/2856 [==============================] - 80s 28ms/step - loss: 0.0334 - accuracy: 0.9907 - f1_m: 0.9904 - precision_m: 0.9911 - recall_m: 0.9900 - val_loss: 0.0606 - val_accuracy: 0.9825 - val_f1_m: 0.9822 - val_precision_m: 0.9832 - val_recall_m: 0.9817 - lr: 0.0100
Epoch 6/50
2856/2856 [==============================] - 81s 28ms/step - loss: 0.0278 - accuracy: 0.9918 - f1_m: 0.9916 - precision_m: 0.9923 - recall_m: 0.9912 - val_loss: 0.0385 - val_accuracy: 0.9886 - val_f1_m: 0.9881 - val_precision_m: 0.9886 - val_recall_m: 0.9878 - lr: 0.0100
Epoch 7/50
```

In [75]:
```
dl_acc = hist8.history["val_accuracy"][28]
dl_prec = hist8.history["val_precision_m"][28]
dl_rec = hist8.history["val_recall_m"][28]
dl_f1 = hist8.history["val_f1_m"][28]

storeResults('TL - Xception',dl_acc,dl_prec,dl_rec,dl_f1)
```

In [47]:
```
model8.save('models/xception.h5')
```

In [48]:
```
x=hist8
plt.figure(figsize=(20,10))
plt.subplot(1, 2, 1)
plt.suptitle('Optimizer : adam', fontsize=10)
plt.ylabel('Loss', fontsize=16)
plt.plot(x.history['loss'], label='Training Loss')
plt.plot(x.history['val_loss'], label='Validation Loss')
plt.legend(loc='upper right')

plt.subplot(1, 2, 2)
plt.ylabel('Accuracy', fontsize=16)
plt.plot(x.history['accuracy'], label='Training Accuracy')
plt.plot(x.history['val_accuracy'], label='Validation Accuracy')
plt.legend(loc='lower right')
plt.show()
```



NASNetMobile

In [49]:
```
# Defining the pretrained base model
base = NASNetMobile(include_top=False, weights='imagenet', input_shape=(256,256,3))
x = base.output
x = GlobalAveragePooling2D()(x)
# Defining the head of the model where the prediction is conducted
head = Dense(4, activation='softmax')(x)
# Combining base and head
model9 = Model(inputs=base.input, outputs=head)
model9.compile(optimizer='sgd',
              loss = 'categorical_crossentropy',
              metrics=["accuracy",f1_m,precision_m, recall_m])
model9.summary()
```

```
Model: "model_8"

 Layer (type)                   Output Shape         Param #     Connected to
==================================================================================================
 input_9 (InputLayer)           [(None, 256, 256, 3  0           []
                                )]

 stem_conv1 (Conv2D)            (None, 127, 127, 32  864         ['input_9[0][0]']
                                )

 stem_bn1 (BatchNormalization)  (None, 127, 127, 32  128         ['stem_conv1[0][0]']
                                )

 activation_297 (Activation)    (None, 127, 127, 32  0           ['stem_bn1[0][0]']
                                )

 reduction_conv_1_stem_1 (Conv2  (None, 127, 127, 11  352        ['activation_297[0][0]']
 D)                             )
```

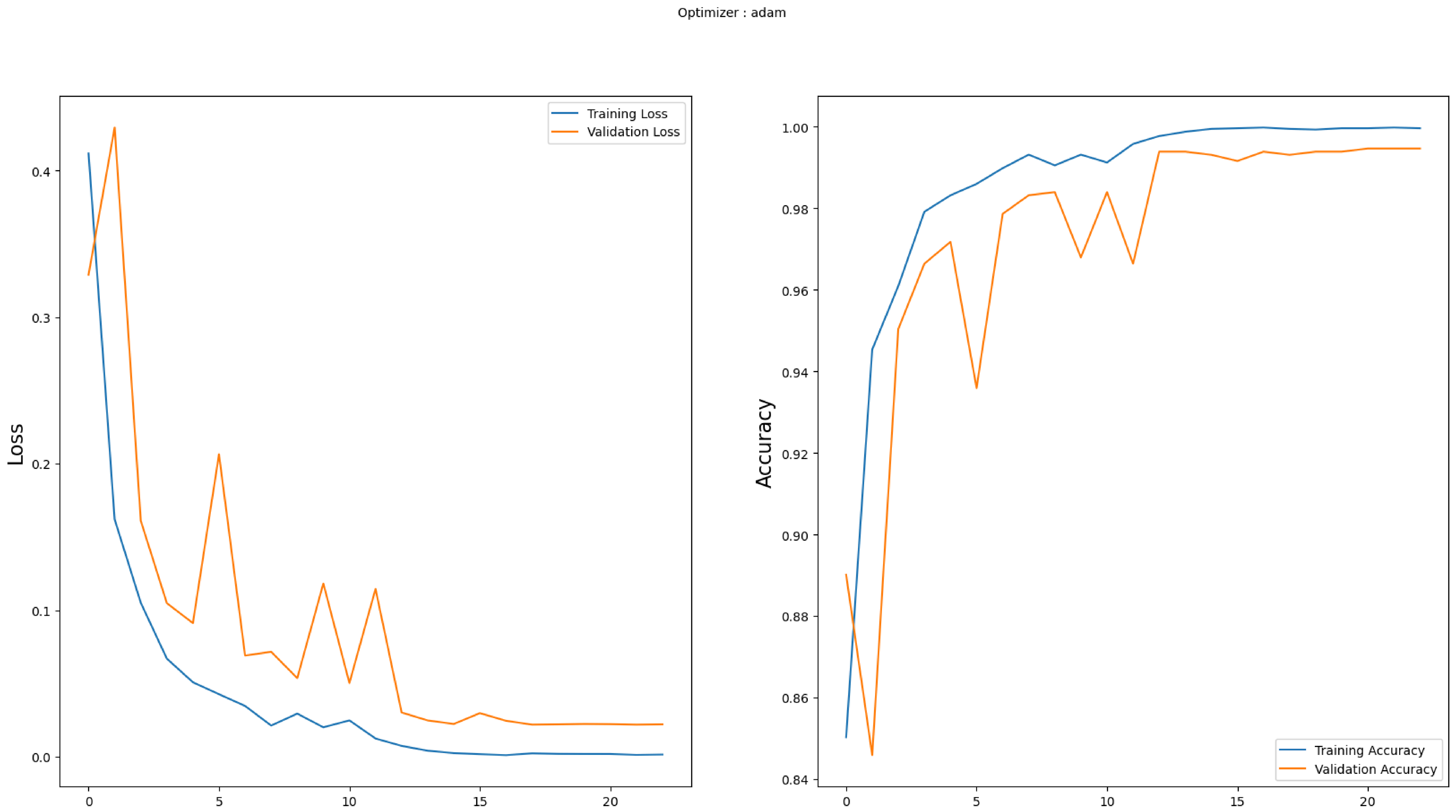
In [50]:
```
hist9 = model9.fit(train_set, epochs=50, validation_data=test_set,steps_per_epoch=len(train_set), validation_steps=len(test_set),callbacks=[learning_rate_reduction, early_stop])
```

```
Epoch 1/50
2856/2856 [==============================] - 289s 95ms/step - loss: 0.4118 - accuracy: 0.8503 - f1_m: 0.8386 - precision_m: 0.8708 - recall_m: 0.8225 - val_loss: 0.3290 - val_ac
curacy: 0.8902 - val_f1_m: 0.8786 - val_precision_m: 0.9040 - val_recall_m: 0.8659 - lr: 0.0100
Epoch 2/50
2856/2856 [==============================] - 264s 92ms/step - loss: 0.1619 - accuracy: 0.9454 - f1_m: 0.9445 - precision_m: 0.9519 - recall_m: 0.9408 - val_loss: 0.4295 - val_ac
curacy: 0.8459 - val_f1_m: 0.8280 - val_precision_m: 0.8636 - val_recall_m: 0.8102 - lr: 0.0100
Epoch 3/50
2856/2856 [==============================] - 265s 93ms/step - loss: 0.1051 - accuracy: 0.9610 - f1_m: 0.9611 - precision_m: 0.9648 - recall_m: 0.9592 - val_loss: 0.1611 - val_ac
curacy: 0.9504 - val_f1_m: 0.9512 - val_precision_m: 0.9543 - val_recall_m: 0.9497 - lr: 0.0100
Epoch 4/50
2856/2856 [==============================] - 265s 93ms/step - loss: 0.0669 - accuracy: 0.9792 - f1_m: 0.9789 - precision_m: 0.9809 - recall_m: 0.9779 - val_loss: 0.1048 - val_ac
curacy: 0.9664 - val_f1_m: 0.9665 - val_precision_m: 0.9680 - val_recall_m: 0.9657 - lr: 0.0100
Epoch 5/50
2856/2856 [==============================] - 265s 93ms/step - loss: 0.0508 - accuracy: 0.9832 - f1_m: 0.9831 - precision_m: 0.9842 - recall_m: 0.9825 - val_loss: 0.0911 - val_ac
curacy: 0.9718 - val_f1_m: 0.9688 - val_precision_m: 0.9764 - val_recall_m: 0.9649 - lr: 0.0100
Epoch 6/50
2856/2856 [==============================] - 264s 92ms/step - loss: 0.0426 - accuracy: 0.9860 - f1_m: 0.9859 - precision_m: 0.9867 - recall_m: 0.9855 - val_loss: 0.2065 - val_ac
curacy: 0.9359 - val_f1_m: 0.9357 - val_precision_m: 0.9413 - val_recall_m: 0.9329 - lr: 0.0100
Epoch 7/50
2856/2856 [==============================] - 265s 93ms/step - loss: 0.0346 - accuracy: 0.9898 - f1_m: 0.9898 - precision_m: 0.9902 - recall_m: 0.9897 - val_loss: 0.0690 - val_ac
curacy: 0.9786 - val_f1_m: 0.9776 - val_precision_m: 0.9802 - val_recall_m: 0.9764 - lr: 0.0100
Epoch 8/50
2856/2856 [==============================] - 265s 93ms/step - loss: 0.0212 - accuracy: 0.9932 - f1_m: 0.9931 - precision_m: 0.9935 - recall_m: 0.9928 - val_loss: 0.0716 - val_ac
curacy: 0.9832 - val_f1_m: 0.9835 - val_precision_m: 0.9840 - val_recall_m: 0.9832 - lr: 0.0100
Epoch 9/50
2856/2856 [==============================] - 264s 93ms/step - loss: 0.0294 - accuracy: 0.9905 - f1_m: 0.9904 - precision_m: 0.9907 - recall_m: 0.9902 - val_loss: 0.0536 - val_ac
curacy: 0.9840 - val_f1_m: 0.9842 - val_precision_m: 0.9848 - val_recall_m: 0.9840 - lr: 0.0100
Epoch 10/50
2856/2856 [==============================] - 265s 93ms/step - loss: 0.0200 - accuracy: 0.9932 - f1_m: 0.9931 - precision_m: 0.9933 - recall_m: 0.9930 - val_loss: 0.1182 - val_ac
curacy: 0.9680 - val_f1_m: 0.9688 - val_precision_m: 0.9703 - val_recall_m: 0.9680 - lr: 0.0100
Epoch 11/50
2856/2856 [==============================] - 265s 93ms/step - loss: 0.0247 - accuracy: 0.9912 - f1_m: 0.9915 - precision_m: 0.9921 - recall_m: 0.9912 - val_loss: 0.0503 - val_ac
curacy: 0.9840 - val_f1_m: 0.9848 - val_precision_m: 0.9863 - val_recall_m: 0.9840 - lr: 0.0100
Epoch 12/50
2856/2856 [==============================] - ETA: 0s - loss: 0.0124 - accuracy: 0.9958 - f1_m: 0.9958 - precision_m: 0.9958 - recall_m: 0.9958
Epoch 12: ReduceLROnPlateau reducing learning rate to 0.0029999999329447745.
2856/2856 [==============================] - 265s 93ms/step - loss: 0.0124 - accuracy: 0.9958 - f1_m: 0.9958 - precision_m: 0.9958 - recall_m: 0.9958 - val_loss: 0.1145 - val_ac
curacy: 0.9664 - val_f1_m: 0.9654 - val_precision_m: 0.9665 - val_recall_m: 0.9649 - lr: 0.0100
Epoch 13/50
2856/2856 [==============================] - 264s 93ms/step - loss: 0.0073 - accuracy: 0.9977 - f1_m: 0.9977 - precision_m: 0.9979 - recall_m: 0.9975 - val_loss: 0.0301 - val_ac
curacy: 0.9939 - val_f1_m: 0.9942 - val_precision_m: 0.9947 - val_recall_m: 0.9939 - lr: 0.0030
Epoch 14/50
2856/2856 [==============================] - 265s 93ms/step - loss: 0.0041 - accuracy: 0.9988 - f1_m: 0.9987 - precision_m: 0.9988 - recall_m: 0.9986 - val_loss: 0.0247 - val_ac
curacy: 0.9939 - val_f1_m: 0.9942 - val_precision_m: 0.9947 - val_recall_m: 0.9939 - lr: 0.0030
Epoch 15/50
2856/2856 [==============================] - 265s 93ms/step - loss: 0.0024 - accuracy: 0.9995 - f1_m: 0.9995 - precision_m: 0.9995 - recall_m: 0.9995 - val_loss: 0.0223 - val_ac
curacy: 0.9931 - val_f1_m: 0.9931 - val_precision_m: 0.9931 - val_recall_m: 0.9931 - lr: 0.0030
Epoch 16/50
2856/2856 [==============================] - ETA: 0s - loss: 0.0017 - accuracy: 0.9996 - f1_m: 0.9996 - precision_m: 0.9996 - recall_m: 0.9996
Epoch 16: ReduceLROnPlateau reducing learning rate to 0.0009000000078231095.
2856/2856 [==============================] - 265s 93ms/step - loss: 0.0017 - accuracy: 0.9996 - f1_m: 0.9996 - precision_m: 0.9996 - recall_m: 0.9996 - val_loss: 0.0297 - val_ac
curacy: 0.9916 - val_f1_m: 0.9916 - val_precision_m: 0.9916 - val_recall_m: 0.9916 - lr: 0.0030
Epoch 17/50
2856/2856 [==============================] - 265s 93ms/step - loss: 9.9940e-04 - accuracy: 0.9998 - f1_m: 0.9998 - precision_m: 0.9998 - recall_m: 0.9998 - val_loss: 0.0245 - va
l_accuracy: 0.9939 - val_f1_m: 0.9939 - val_precision_m: 0.9939 - val_recall_m: 0.9939 - lr: 9.0000e-04
Epoch 18/50
2856/2856 [==============================] - 264s 93ms/step - loss: 0.0022 - accuracy: 0.9995 - f1_m: 0.9995 - precision_m: 0.9995 - recall_m: 0.9995 - val_loss: 0.0218 - val_ac
curacy: 0.9931 - val_f1_m: 0.9931 - val_precision_m: 0.9931 - val_recall_m: 0.9931 - lr: 9.0000e-04
Epoch 19/50
2856/2856 [==============================] - ETA: 0s - loss: 0.0019 - accuracy: 0.9993 - f1_m: 0.9993 - precision_m: 0.9993 - recall_m: 0.9993
Epoch 19: ReduceLROnPlateau reducing learning rate to 0.00026999999536201356.
2856/2856 [==============================] - 265s 93ms/step - loss: 0.0019 - accuracy: 0.9993 - f1_m: 0.9993 - precision_m: 0.9993 - recall_m: 0.9993 - val_loss: 0.0220 - val_ac
curacy: 0.9939 - val_f1_m: 0.9934 - val_precision_m: 0.9939 - val_recall_m: 0.9931 - lr: 9.0000e-04
Epoch 20/50
2856/2856 [==============================] - 265s 93ms/step - loss: 0.0018 - accuracy: 0.9996 - f1_m: 0.9996 - precision_m: 0.9996 - recall_m: 0.9996 - val_loss: 0.0223 - val_ac
curacy: 0.9939 - val_f1_m: 0.9934 - val_precision_m: 0.9939 - val_recall_m: 0.9931 - lr: 2.7000e-04
Epoch 21/50
2856/2856 [==============================] - 265s 93ms/step - loss: 0.0018 - accuracy: 0.9996 - f1_m: 0.9996 - precision_m: 0.9996 - recall_m: 0.9996 - val_loss: 0.0222 - val_ac
curacy: 0.9947 - val_f1_m: 0.9942 - val_precision_m: 0.9947 - val_recall_m: 0.9939 - lr: 2.7000e-04
Epoch 22/50
2856/2856 [==============================] - 264s 93ms/step - loss: 0.0012 - accuracy: 0.9998 - f1_m: 0.9998 - precision_m: 0.9998 - recall_m: 0.9998 - val_loss: 0.0218 - val_ac
curacy: 0.9947 - val_f1_m: 0.9942 - val_precision_m: 0.9947 - val_recall_m: 0.9939 - lr: 2.7000e-04
Epoch 23/50
2856/2856 [==============================] - ETA: 0s - loss: 0.0014 - accuracy: 0.9996 - f1_m: 0.9996 - precision_m: 0.9996 - recall_m: 0.9996Restoring model weights from the en
d of the best epoch: 13.
2856/2856 [==============================] - 265s 93ms/step - loss: 0.0014 - accuracy: 0.9996 - f1_m: 0.9996 - precision_m: 0.9996 - recall_m: 0.9996 - val_loss: 0.0220 - val_ac
curacy: 0.9947 - val_f1_m: 0.9942 - val_precision_m: 0.9947 - val_recall_m: 0.9939 - lr: 2.7000e-04
Epoch 23: early stopping
```

In [76]:
```
dl_acc = hist9.history["val_accuracy"][22]
dl_prec = hist9.history["val_precision_m"][22]
dl_rec = hist9.history["val_recall_m"][22]
dl_f1 = hist9.history["val_f1_m"][22]

storeResults('TL - NASNetMobile',dl_acc,dl_prec,dl_rec,dl_f1)
```

In [51]:
```
model9.save('models/nasnetmobile.h5')
```

In [52]:
```
x=hist9
plt.figure(figsize=(20,10))
plt.subplot(1, 2, 1)
plt.suptitle('Optimizer : adam', fontsize=10)
plt.ylabel('Loss', fontsize=16)
plt.plot(x.history['loss'], label='Training Loss')
plt.plot(x.history['val_loss'], label='Validation Loss')
plt.legend(loc='upper right')

plt.subplot(1, 2, 2)
plt.ylabel('Accuracy', fontsize=16)
plt.plot(x.history['accuracy'], label='Training Accuracy')
plt.plot(x.history['val_accuracy'], label='Validation Accuracy')
plt.legend(loc='lower right')
plt.show()
```



NASNetLarge

In [53]:
```
# Defining the pretrained base model
base = NASNetLarge(include_top=False, weights='imagenet', input_shape=(256,256,3))
x = base.output
x = GlobalAveragePooling2D()(x)
# Defining the head of the model where the prediction is conducted
head = Dense(4, activation='softmax')(x)
# Combining base and head
model10 = Model(inputs=base.input, outputs=head)
model10.compile(optimizer='sgd',
              loss = 'categorical_crossentropy',
              metrics=["accuracy",f1_m,precision_m, recall_m])
model10.summary()
```

```
Model: "model_9"
__________________________________________________________________________________________________
 Layer (type)                   Output Shape         Param #     Connected to
==================================================================================================
 input_10 (InputLayer)          [(None, 256, 256, 3  0           []
                                )]

 stem_conv1 (Conv2D)            (None, 127, 127, 96  2592        ['input_10[0][0]']
                                )

 stem_bn1 (BatchNormalization)  (None, 127, 127, 96  384         ['stem_conv1[0][0]']
                                )

 activation_485 (Activation)    (None, 127, 127, 96  0           ['stem_bn1[0][0]']
                                )

 reduction_conv_1_stem_1 (Conv2  (None, 127, 127, 42  4032       ['activation_485[0][0]']
 D)                             )
```

In [54]:
```
hist10 = model10.fit(train_set, epochs=50, validation_data=test_set,steps_per_epoch=len(train_set), validation_steps=len(test_set),callbacks=[learning_rate_reduction, early_stop])
```

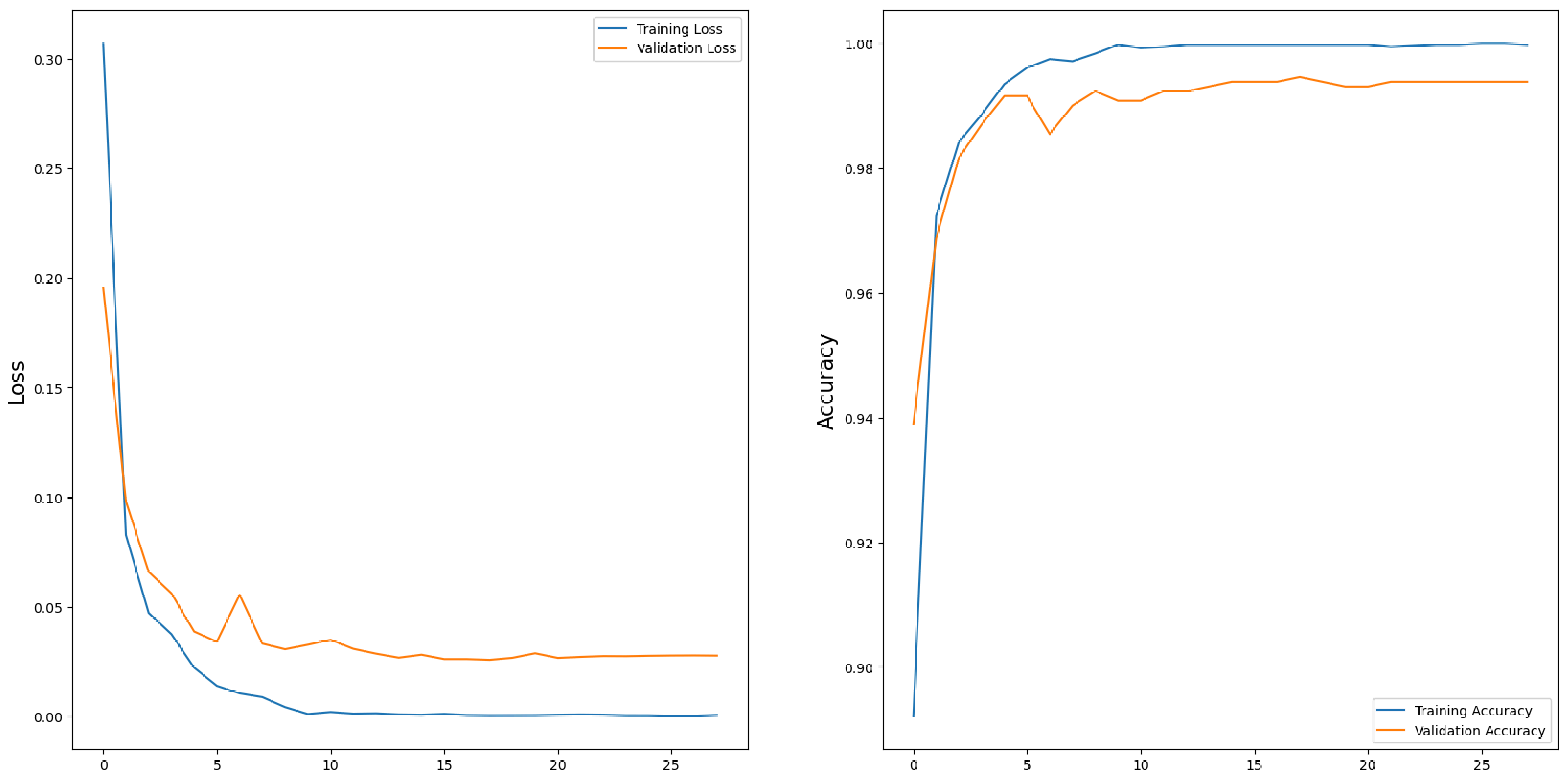
```
Epoch 1/50
2856/2856 [==============================] - 395s 130ms/step - loss: 0.3068 - accuracy: 0.8922 - f1_m: 0.8748 - precision_m: 0.9018 - recall_m: 0.8613 - val_loss: 0.1955 - val_accuracy: 0.9390 - val_f1_m: 0.9400 - val_precision_m: 0.9436 - val_recall_m: 0.9383 - lr: 0.0100
Epoch 2/50
2856/2856 [==============================] - 366s 128ms/step - loss: 0.0828 - accuracy: 0.9723 - f1_m: 0.9723 - precision_m: 0.9764 - recall_m: 0.9702 - val_loss: 0.0982 - val_accuracy: 0.9687 - val_f1_m: 0.9690 - val_precision_m: 0.9756 - val_recall_m: 0.9657 - lr: 0.0100
Epoch 3/50
2856/2856 [==============================] - 366s 128ms/step - loss: 0.0473 - accuracy: 0.9842 - f1_m: 0.9833 - precision_m: 0.9853 - recall_m: 0.9823 - val_loss: 0.0660 - val_accuracy: 0.9817 - val_f1_m: 0.9817 - val_precision_m: 0.9863 - val_recall_m: 0.9794 - lr: 0.0100
Epoch 4/50
2856/2856 [==============================] - 366s 128ms/step - loss: 0.0376 - accuracy: 0.9886 - f1_m: 0.9889 - precision_m: 0.9897 - recall_m: 0.9884 - val_loss: 0.0562 - val_accuracy: 0.9870 - val_f1_m: 0.9876 - val_precision_m: 0.9901 - val_recall_m: 0.9863 - lr: 0.0100
Epoch 5/50
2856/2856 [==============================] - 366s 128ms/step - loss: 0.0223 - accuracy: 0.9935 - f1_m: 0.9935 - precision_m: 0.9939 - recall_m: 0.9933 - val_loss: 0.0388 - val_accuracy: 0.9916 - val_f1_m: 0.9916 - val_precision_m: 0.9916 - val_recall_m: 0.9916 - lr: 0.0100
Epoch 6/50
2856/2856 [==============================] - 365s 128ms/step - loss: 0.0140 - accuracy: 0.9961 - f1_m: 0.9960 - precision_m: 0.9961 - recall_m: 0.9960 - val_loss: 0.0341 - val_accuracy: 0.9916 - val_f1_m: 0.9916 - val_precision_m: 0.9916 - val_recall_m: 0.9916 - lr: 0.0100
Epoch 7/50
```

In [77]:
```
dl_acc = hist10.history["val_accuracy"][27]
dl_prec = hist10.history["val_precision_m"][27]
dl_rec = hist10.history["val_recall_m"][27]
dl_f1 = hist10.history["val_f1_m"][27]

storeResults('TL - NASNetLarge',dl_acc,dl_prec,dl_rec,dl_f1)
```

In [55]:
```
model10.save('models/nasnetlarge.h5')
```

In [56]:
```
x=hist10
plt.figure(figsize=(20,10))
plt.subplot(1, 2, 1)
plt.suptitle('Optimizer : adam', fontsize=10)
plt.ylabel('Loss', fontsize=16)
plt.plot(x.history['loss'], label='Training Loss')
plt.plot(x.history['val_loss'], label='Validation Loss')
plt.legend(loc='upper right')

plt.subplot(1, 2, 2)
plt.ylabel('Accuracy', fontsize=16)
plt.plot(x.history['accuracy'], label='Training Accuracy')
plt.plot(x.history['val_accuracy'], label='Validation Accuracy')
plt.legend(loc='lower right')
plt.show()
```

Optimizer : adam



IVX16

In [57]:
```
from tensorflow.keras.models import load_model
from tensorflow.keras.layers import Average
import efficientnet.keras
```

In [58]:
```
def ensemble():

    model_1 = load_model('models/inceptionv3.h5', compile=False)
    model_1 = Model(inputs = model_1.inputs, outputs = model_1.outputs, name = 'InceptionV3')

    model_2 = load_model('models/xception.h5', compile=False)
    model_2 = Model(inputs = model_2.inputs, outputs = model_2.outputs, name = 'Xception')

    model_3 = load_model('models/vgg16.h5', compile=False)
    model_3 = Model(inputs = model_3.inputs, outputs = model_3.outputs, name = 'VGG16')

    models = [model_1, model_2, model_3]


    models_input = Input(shape =(256,256,3))
    models_output = [model(models_input) for model in models]

    ensemble_output = Average()(models_output)

    simple_average = Model(inputs = models_input, outputs = ensemble_output, name = 'IVX16')

    return simple_average
```

In [59]:
```
model = ensemble()
model.compile(optimizer='sgd',
              loss = 'categorical_crossentropy',
              metrics=["accuracy",f1_m,precision_m, recall_m])
model.summary()
```

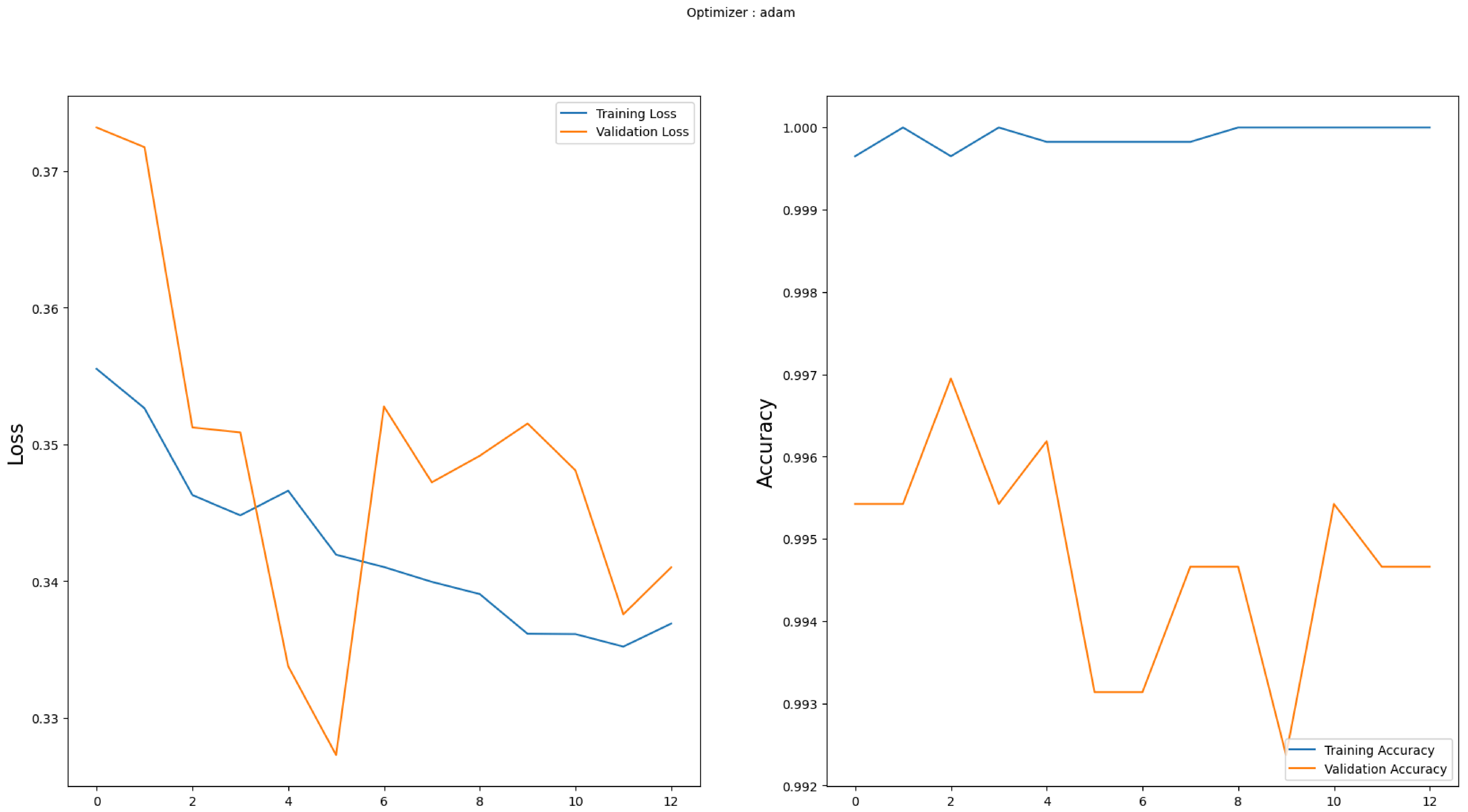
```
Model: "IVX16"
__________________________________________________________________________________________________
 Layer (type)                   Output Shape         Param #     Connected to                     
==================================================================================================
 input_11 (InputLayer)          [(None, 256, 256, 3  0           []                               
                                )]                                                                
                                                                                                  
 InceptionV3 (Functional)       (None, 4)            23855788    ['input_11[0][0]']               
                                                                                                  
 Xception (Functional)          (None, 4)            20869676    ['input_11[0][0]']               
                                                                                                  
 average (Average)              (None, 4)            0           ['InceptionV3[0][0]',            
                                                                  'Xception[0][0]']               
                                                                                                  
==================================================================================================
Total params: 44,725,464
Trainable params: 44,636,504
Non-trainable params: 88,960
__________________________________________________________________________________________________
```

In [60]:
```
history = model.fit(
    train_set,
    epochs=50,
    validation_data=test_set,callbacks=[learning_rate_reduction, early_stop])
```

```
Epoch 1/50
2856/2856 [==============================] - 224s 75ms/step - loss: 0.3555 - accuracy: 0.9996 - f1_m: 0.9992 - precision_m: 1.0000 - recall_m: 0.9988 - val_loss: 0.3732 - val_ac
curacy: 0.9954 - val_f1_m: 0.9936 - val_precision_m: 0.9962 - val_recall_m: 0.9924 - lr: 0.0100
Epoch 2/50
2856/2856 [==============================] - 214s 75ms/step - loss: 0.3526 - accuracy: 1.0000 - f1_m: 1.0000 - precision_m: 1.0000 - recall_m: 1.0000 - val_loss: 0.3717 - val_ac
curacy: 0.9954 - val_f1_m: 0.9949 - val_precision_m: 0.9970 - val_recall_m: 0.9939 - lr: 0.0100
Epoch 3/50
2856/2856 [==============================] - 213s 75ms/step - loss: 0.3463 - accuracy: 0.9996 - f1_m: 0.9996 - precision_m: 0.9998 - recall_m: 0.9995 - val_loss: 0.3512 - val_ac
curacy: 0.9969 - val_f1_m: 0.9931 - val_precision_m: 0.9977 - val_recall_m: 0.9909 - lr: 0.0100
Epoch 4/50
2856/2856 [==============================] - 213s 75ms/step - loss: 0.3448 - accuracy: 1.0000 - f1_m: 0.9999 - precision_m: 1.0000 - recall_m: 0.9998 - val_loss: 0.3509 - val_ac
curacy: 0.9954 - val_f1_m: 0.9936 - val_precision_m: 0.9962 - val_recall_m: 0.9924 - lr: 0.0100
Epoch 5/50
2856/2856 [==============================] - 214s 75ms/step - loss: 0.3466 - accuracy: 0.9998 - f1_m: 0.9998 - precision_m: 1.0000 - recall_m: 0.9996 - val_loss: 0.3338 - val_ac
curacy: 0.9962 - val_f1_m: 0.9954 - val_precision_m: 0.9970 - val_recall_m: 0.9947 - lr: 0.0100
Epoch 6/50
2856/2856 [==============================] - ETA: 0s - loss: 0.3419 - accuracy: 0.9998 - f1_m: 0.9994 - precision_m: 1.0000 - recall_m: 0.9991
Epoch 6: ReduceLROnPlateau reducing learning rate to 0.0029999999329447745.
2856/2856 [==============================] - 213s 74ms/step - loss: 0.3419 - accuracy: 0.9998 - f1_m: 0.9994 - precision_m: 1.0000 - recall_m: 0.9991 - val_loss: 0.3273 - val_ac
curacy: 0.9931 - val_f1_m: 0.9942 - val_precision_m: 0.9962 - val_recall_m: 0.9931 - lr: 0.0100
Epoch 7/50
2856/2856 [==============================] - 213s 75ms/step - loss: 0.3410 - accuracy: 0.9998 - f1_m: 0.9999 - precision_m: 1.0000 - recall_m: 0.9998 - val_loss: 0.3528 - val_ac
curacy: 0.9931 - val_f1_m: 0.9934 - val_precision_m: 0.9970 - val_recall_m: 0.9916 - lr: 0.0030
Epoch 8/50
2856/2856 [==============================] - 214s 75ms/step - loss: 0.3399 - accuracy: 0.9998 - f1_m: 0.9997 - precision_m: 0.9998 - recall_m: 0.9996 - val_loss: 0.3472 - val_ac
curacy: 0.9947 - val_f1_m: 0.9944 - val_precision_m: 0.9954 - val_recall_m: 0.9939 - lr: 0.0030
Epoch 9/50
2856/2856 [==============================] - ETA: 0s - loss: 0.3391 - accuracy: 1.0000 - f1_m: 0.9996 - precision_m: 1.0000 - recall_m: 0.9995
Epoch 9: ReduceLROnPlateau reducing learning rate to 0.0009000000078231095.
2856/2856 [==============================] - 213s 75ms/step - loss: 0.3391 - accuracy: 1.0000 - f1_m: 0.9996 - precision_m: 1.0000 - recall_m: 0.9995 - val_loss: 0.3492 - val_ac
curacy: 0.9947 - val_f1_m: 0.9944 - val_precision_m: 0.9970 - val_recall_m: 0.9931 - lr: 0.0030
Epoch 10/50
2856/2856 [==============================] - 214s 75ms/step - loss: 0.3361 - accuracy: 1.0000 - f1_m: 0.9996 - precision_m: 1.0000 - recall_m: 0.9995 - val_loss: 0.3515 - val_ac
curacy: 0.9924 - val_f1_m: 0.9931 - val_precision_m: 0.9962 - val_recall_m: 0.9916 - lr: 9.0000e-04
Epoch 11/50
2856/2856 [==============================] - 214s 75ms/step - loss: 0.3361 - accuracy: 1.0000 - f1_m: 0.9996 - precision_m: 1.0000 - recall_m: 0.9995 - val_loss: 0.3481 - val_ac
curacy: 0.9954 - val_f1_m: 0.9944 - val_precision_m: 0.9970 - val_recall_m: 0.9931 - lr: 9.0000e-04
Epoch 12/50
2856/2856 [==============================] - ETA: 0s - loss: 0.3352 - accuracy: 1.0000 - f1_m: 1.0000 - precision_m: 1.0000 - recall_m: 1.0000
Epoch 12: ReduceLROnPlateau reducing learning rate to 0.00026999999536201356.
2856/2856 [==============================] - 213s 75ms/step - loss: 0.3352 - accuracy: 1.0000 - f1_m: 1.0000 - precision_m: 1.0000 - recall_m: 1.0000 - val_loss: 0.3376 - val_ac
curacy: 0.9947 - val_f1_m: 0.9954 - val_precision_m: 0.9970 - val_recall_m: 0.9947 - lr: 9.0000e-04
Epoch 13/50
2856/2856 [==============================] - ETA: 0s - loss: 0.3369 - accuracy: 1.0000 - f1_m: 0.9998 - precision_m: 1.0000 - recall_m: 0.9996Restoring model weights from the en
d of the best epoch: 3.
2856/2856 [==============================] - 214s 75ms/step - loss: 0.3369 - accuracy: 1.0000 - f1_m: 0.9998 - precision_m: 1.0000 - recall_m: 0.9996 - val_loss: 0.3410 - val_ac
curacy: 0.9947 - val_f1_m: 0.9936 - val_precision_m: 0.9962 - val_recall_m: 0.9924 - lr: 2.7000e-04
Epoch 13: early stopping
```

In [78]:
```
dl_acc = history.history["val_accuracy"][12]
dl_prec = history.history["val_precision_m"][12]
dl_rec = history.history["val_recall_m"][12]
dl_f1 = history.history["val_f1_m"][12]

storeResults('TL - IVX16',dl_acc,dl_prec,dl_rec,dl_f1)
```

In [61]:
```
x=history
plt.figure(figsize=(20,10))
plt.subplot(1, 2, 1)
plt.suptitle('Optimizer : adam', fontsize=10)
plt.ylabel('Loss', fontsize=16)
plt.plot(x.history['loss'], label='Training Loss')
plt.plot(x.history['val_loss'], label='Validation Loss')
plt.legend(loc='upper right')

plt.subplot(1, 2, 2)
plt.ylabel('Accuracy', fontsize=16)
plt.plot(x.history['accuracy'], label='Training Accuracy')
plt.plot(x.history['val_accuracy'], label='Validation Accuracy')
plt.legend(loc='lower right')
plt.show()
```



Extension

In [62]:
```
def ensemble():

    model_1 = load_model('models/nasnetlarge.h5', compile=False)
    model_1 = Model(inputs = model_1.inputs, outputs = model_1.outputs, name = 'NASNetLarge')

    model_2 = load_model('models/nasnetmobile.h5', compile=False)
    model_2 = Model(inputs = model_2.inputs, outputs = model_2.outputs, name = 'NASNetMobile')

    models = [model_1, model_2]


    models_input = Input(shape =(256,256,3))
    models_output = [model(models_input) for model in models]

    ensemble_output = Average()(models_output)

    simple_average = Model(inputs = models_input, outputs = ensemble_output, name = 'Extension')

    return simple_average
```

In [63]:
```
ext = ensemble()
ext.compile(optimizer='sgd',
              loss = 'categorical_crossentropy',
              metrics=["accuracy",f1_m,precision_m, recall_m])
ext.summary()
```

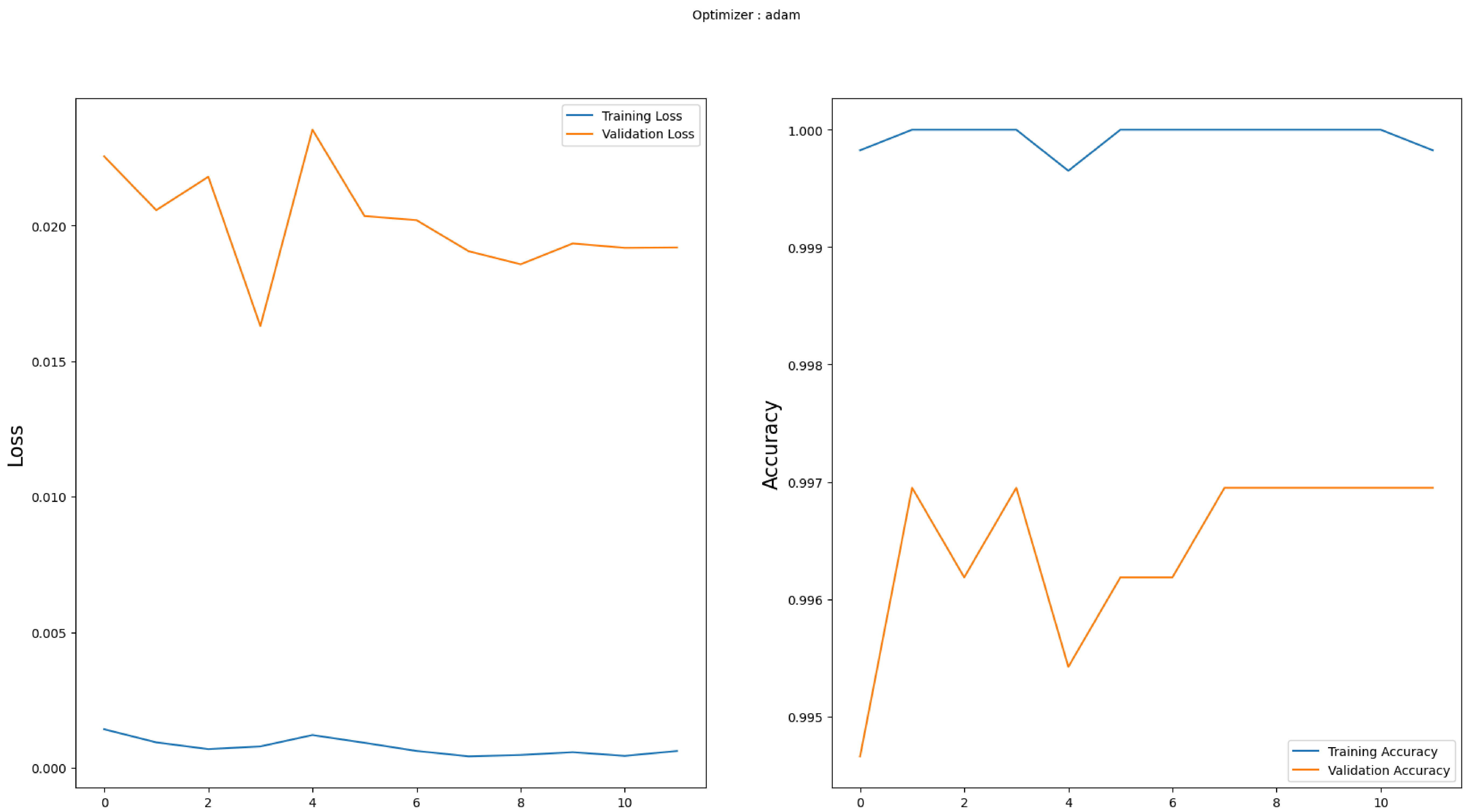
```
Model: "Extension"
__________________________________________________________________________________________________
 Layer (type)                   Output Shape         Param #     Connected to
==================================================================================================
 input_12 (InputLayer)          [(None, 256, 256, 3  0           []
                                )]

 NASNetLarge (Functional)       (None, 4)            84932950    ['input_12[0][0]']

 NASNetMobile (Functional)      (None, 4)            4273944     ['input_12[0][0]']

 average_1 (Average)            (None, 4)            0           ['NASNetLarge[0][0]',
                                                                  'NASNetMobile[0][0]']

==================================================================================================
Total params: 89,206,894
Trainable params: 88,973,488
Non-trainable params: 233,406
__________________________________________________________________________________________________
```

In [64]:
```
history1 = ext.fit(
    train_set,
    epochs=50,
    validation_data=test_set,callbacks=[learning_rate_reduction, early_stop])
```

```
Epoch 1/50
2856/2856 [==============================] - 672s 220ms/step - loss: 0.0014 - accuracy: 0.9998 - f1_m: 0.9998 - precision_m: 0.9998 - recall_m: 0.9998 - val_loss: 0.0225 - val_a
ccuracy: 0.9947 - val_f1_m: 0.9952 - val_precision_m: 0.9962 - val_recall_m: 0.9947 - lr: 0.0100
Epoch 2/50
2856/2856 [==============================] - 618s 216ms/step - loss: 9.3611e-04 - accuracy: 1.0000 - f1_m: 1.0000 - precision_m: 1.0000 - recall_m: 1.0000 - val_loss: 0.0206 - v
al_accuracy: 0.9969 - val_f1_m: 0.9959 - val_precision_m: 0.9970 - val_recall_m: 0.9954 - lr: 0.0100
Epoch 3/50
2856/2856 [==============================] - 617s 216ms/step - loss: 6.8648e-04 - accuracy: 1.0000 - f1_m: 1.0000 - precision_m: 1.0000 - recall_m: 1.0000 - val_loss: 0.0218 - v
al_accuracy: 0.9962 - val_f1_m: 0.9962 - val_precision_m: 0.9962 - val_recall_m: 0.9962 - lr: 0.0100
Epoch 4/50
2856/2856 [==============================] - 617s 216ms/step - loss: 7.8715e-04 - accuracy: 1.0000 - f1_m: 1.0000 - precision_m: 1.0000 - recall_m: 1.0000 - val_loss: 0.0163 - v
al_accuracy: 0.9969 - val_f1_m: 0.9970 - val_precision_m: 0.9970 - val_recall_m: 0.9970 - lr: 0.0100
Epoch 5/50
2856/2856 [==============================] - ETA: 0s - loss: 0.0012 - accuracy: 0.9996 - f1_m: 0.9996 - precision_m: 0.9996 - recall_m: 0.9996
Epoch 5: ReduceLROnPlateau reducing learning rate to 0.0029999999329447745.
2856/2856 [==============================] - 617s 216ms/step - loss: 0.0012 - accuracy: 0.9996 - f1_m: 0.9996 - precision_m: 0.9996 - recall_m: 0.9996 - val_loss: 0.0235 - val_a
ccuracy: 0.9954 - val_f1_m: 0.9952 - val_precision_m: 0.9962 - val_recall_m: 0.9947 - lr: 0.0100
Epoch 6/50
2856/2856 [==============================] - 617s 216ms/step - loss: 9.2013e-04 - accuracy: 1.0000 - f1_m: 1.0000 - precision_m: 1.0000 - recall_m: 1.0000 - val_loss: 0.0203 - v
al_accuracy: 0.9962 - val_f1_m: 0.9959 - val_precision_m: 0.9970 - val_recall_m: 0.9954 - lr: 0.0030
Epoch 7/50
2856/2856 [==============================] - 616s 216ms/step - loss: 6.1913e-04 - accuracy: 1.0000 - f1_m: 1.0000 - precision_m: 1.0000 - recall_m: 1.0000 - val_loss: 0.0202 - v
al_accuracy: 0.9962 - val_f1_m: 0.9959 - val_precision_m: 0.9970 - val_recall_m: 0.9954 - lr: 0.0030
Epoch 8/50
2856/2856 [==============================] - ETA: 0s - loss: 4.1802e-04 - accuracy: 1.0000 - f1_m: 1.0000 - precision_m: 1.0000 - recall_m: 1.0000
Epoch 8: ReduceLROnPlateau reducing learning rate to 0.0009000000078231095.
2856/2856 [==============================] - 616s 216ms/step - loss: 4.1802e-04 - accuracy: 1.0000 - f1_m: 1.0000 - precision_m: 1.0000 - recall_m: 1.0000 - val_loss: 0.0191 - v
al_accuracy: 0.9969 - val_f1_m: 0.9959 - val_precision_m: 0.9970 - val_recall_m: 0.9954 - lr: 0.0030
Epoch 9/50
2856/2856 [==============================] - 616s 216ms/step - loss: 4.6883e-04 - accuracy: 1.0000 - f1_m: 1.0000 - precision_m: 1.0000 - recall_m: 1.0000 - val_loss: 0.0186 - v
al_accuracy: 0.9969 - val_f1_m: 0.9959 - val_precision_m: 0.9970 - val_recall_m: 0.9954 - lr: 9.0000e-04
Epoch 10/50
2856/2856 [==============================] - 615s 215ms/step - loss: 5.7156e-04 - accuracy: 1.0000 - f1_m: 1.0000 - precision_m: 1.0000 - recall_m: 1.0000 - val_loss: 0.0193 - v
al_accuracy: 0.9969 - val_f1_m: 0.9959 - val_precision_m: 0.9970 - val_recall_m: 0.9954 - lr: 9.0000e-04
Epoch 11/50
2856/2856 [==============================] - ETA: 0s - loss: 4.3507e-04 - accuracy: 1.0000 - f1_m: 1.0000 - precision_m: 1.0000 - recall_m: 1.0000
Epoch 11: ReduceLROnPlateau reducing learning rate to 0.00026999999536201356.
2856/2856 [==============================] - 617s 216ms/step - loss: 4.3507e-04 - accuracy: 1.0000 - f1_m: 1.0000 - precision_m: 1.0000 - recall_m: 1.0000 - val_loss: 0.0192 - v
al_accuracy: 0.9969 - val_f1_m: 0.9959 - val_precision_m: 0.9970 - val_recall_m: 0.9954 - lr: 9.0000e-04
Epoch 12/50
2856/2856 [==============================] - ETA: 0s - loss: 6.1644e-04 - accuracy: 0.9998 - f1_m: 0.9998 - precision_m: 0.9998 - recall_m: 0.9998Restoring model weights from th
e end of the best epoch: 2.
2856/2856 [==============================] - 617s 216ms/step - loss: 6.1644e-04 - accuracy: 0.9998 - f1_m: 0.9998 - precision_m: 0.9998 - recall_m: 0.9998 - val_loss: 0.0192 - v
al_accuracy: 0.9969 - val_f1_m: 0.9959 - val_precision_m: 0.9970 - val_recall_m: 0.9954 - lr: 2.7000e-04
Epoch 12: early stopping
```

In [81]:
```
dl_acc = history1.history["val_accuracy"][11]
dl_prec = history1.history["val_precision_m"][11]
dl_rec = history1.history["val_recall_m"][11]
dl_f1 = history1.history["val_f1_m"][11]

storeResults('TL - Extension',dl_acc,dl_prec,dl_rec,dl_f1)
```

In [80]:
```
x=history1
plt.figure(figsize=(20,10))
plt.subplot(1, 2, 1)
plt.suptitle('Optimizer : adam', fontsize=10)
plt.ylabel('Loss', fontsize=16)
plt.plot(x.history['loss'], label='Training Loss')
plt.plot(x.history['val_loss'], label='Validation Loss')
plt.legend(loc='upper right')

plt.subplot(1, 2, 2)
plt.ylabel('Accuracy', fontsize=16)
plt.plot(x.history['accuracy'], label='Training Accuracy')
plt.plot(x.history['val_accuracy'], label='Validation Accuracy')
plt.legend(loc='lower right')
plt.show()
```



Comparison

In [82]:
```
#creating dataframe
import pandas as pd
result = pd.DataFrame({ 'ML Model' : ML_Model,
                        'Accuracy' : accuracy,
                        'Precision': precision,
                        'Recall'   : recall,
                        'F1-Score': f1score,

                      })
```

In [83]:
```
result
```

Out[83]:

	ML Model	Accuracy	Precision	Recall	F1-Score
0	VisionTransformer - SWIN	0.811	0.818	0.797	0.804
1	VisionTransformer - CCT	0.590	0.617	0.495	0.536
2	VisionTransformer - EANet	0.511	0.511	0.504	0.506
3	TL - VGG16	0.309	0.095	0.309	0.146
4	TL - VGG19	0.309	0.095	0.309	0.146
5	TL - InceptionV3	0.477	0.434	0.245	0.308
6	TL - ResNet50	0.665	0.689	0.463	0.539
7	TL - InceptionResNetV2	0.484	0.429	0.247	0.308
8	TL - Xception	0.995	0.995	0.995	0.995
9	TL - NASNetMobile	0.995	0.995	0.994	0.994
10	TL - NASNetLarge	0.994	0.994	0.994	0.994
11	TL - IVX16	0.995	0.996	0.992	0.994
12	TL - Extension	0.997	0.997	0.995	0.996

Modelling

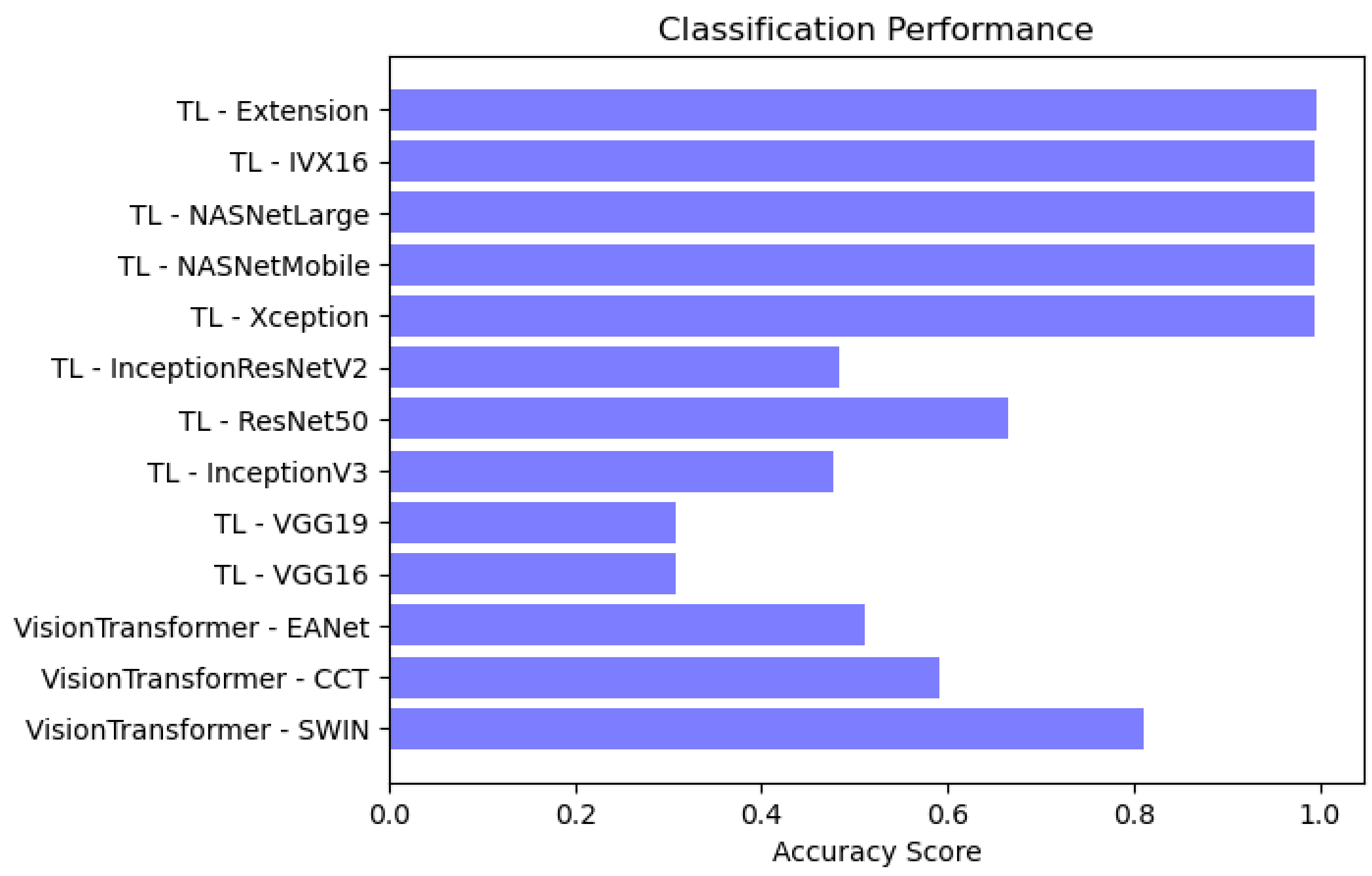
In [84]:
```
ext.save('models/extension.h5')
```

Graph

In [85]:
```
classifier = ML_Model
y_pos = np.arange(len(classifier))
```

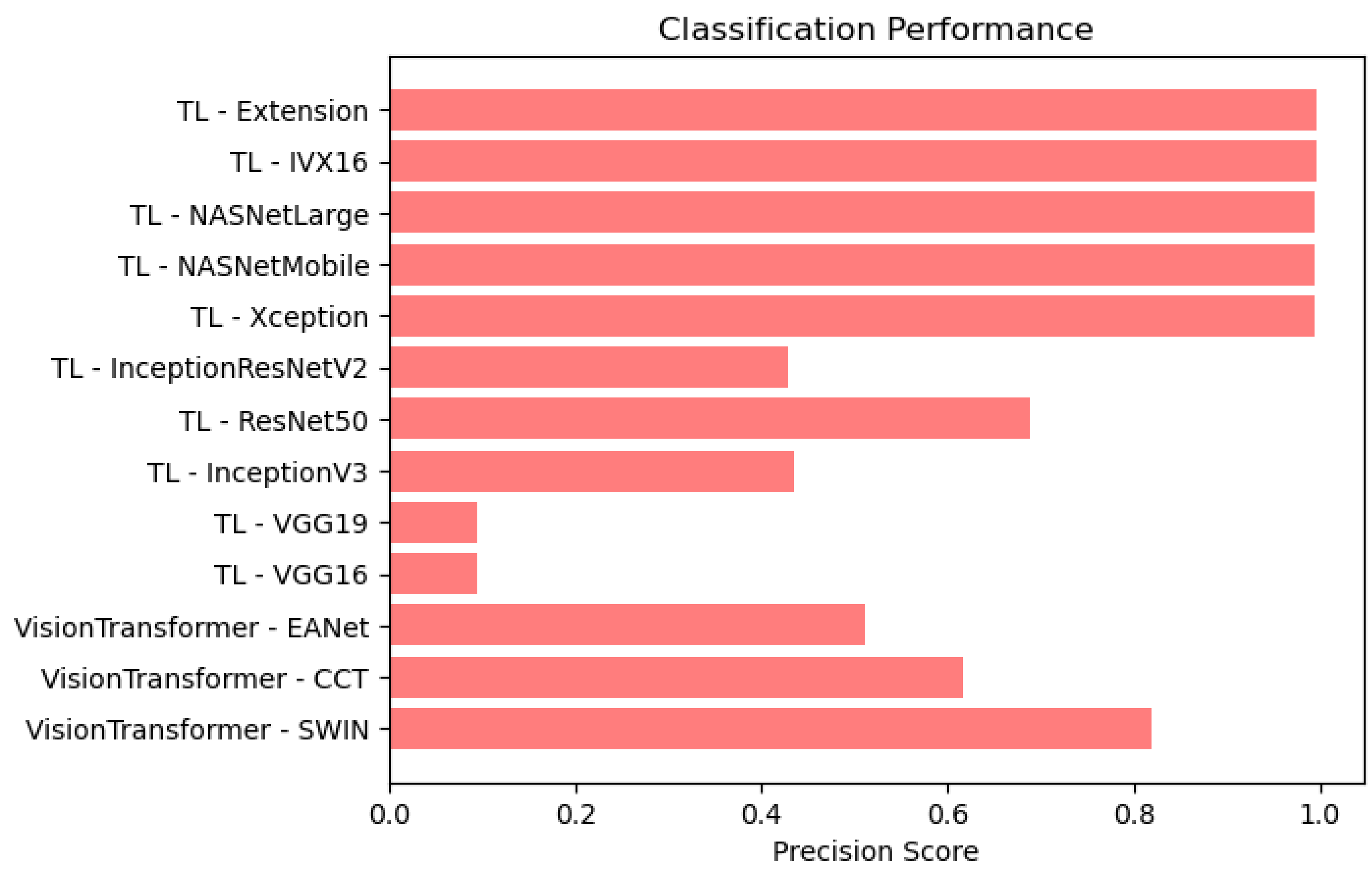
Accuracy

In [86]:
```
import matplotlib.pyplot as plt2
plt2.barh(y_pos, accuracy, align='center', alpha=0.5,color='blue')
plt2.yticks(y_pos, classifier)
plt2.xlabel('Accuracy Score')
plt2.title('Classification Performance')
plt2.show()
```



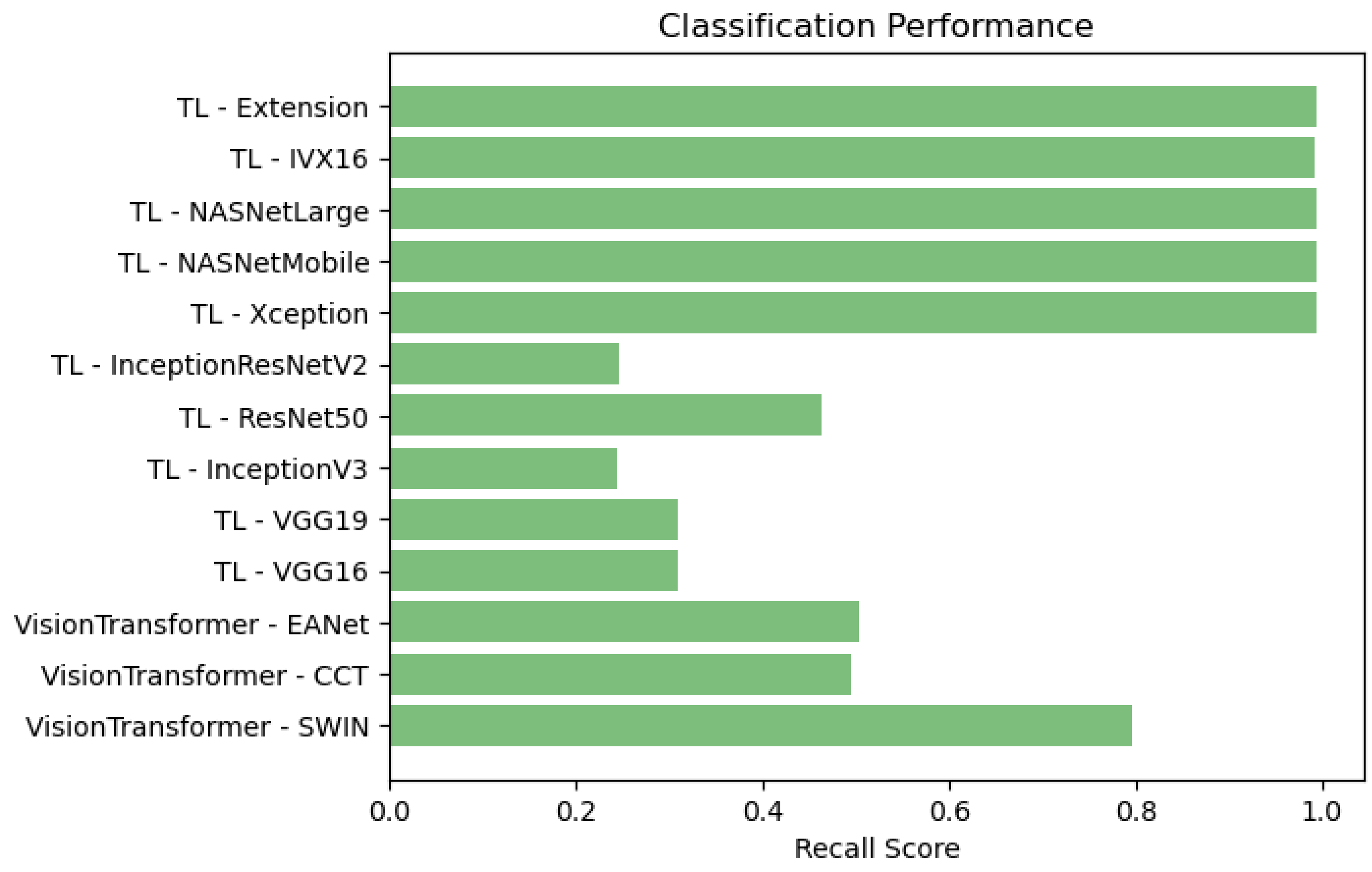
Precision

In [87]:
```
plt2.barh(y_pos, precision, align='center', alpha=0.5,color='red')
plt2.yticks(y_pos, classifier)
plt2.xlabel('Precision Score')
plt2.title('Classification Performance')
plt2.show()
```



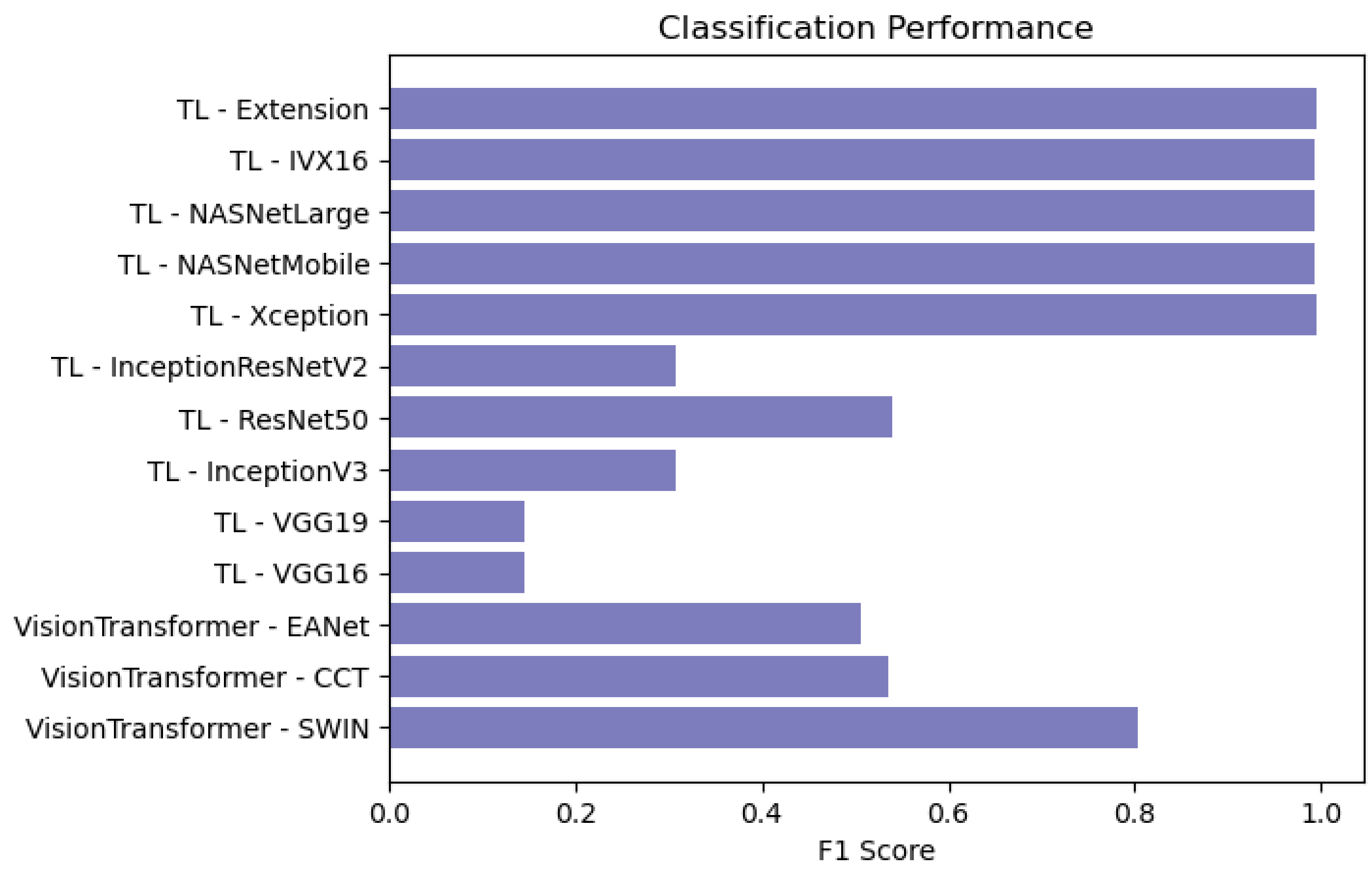
Recall

In [88]:
```
plt2.barh(y_pos, recall, align='center', alpha=0.5,color='green')
plt2.yticks(y_pos, classifier)
plt2.xlabel('Recall Score')
plt2.title('Classification Performance')
plt2.show()
```



F1 Score

In [89]:
```
plt2.barh(y_pos, f1score, align='center', alpha=0.5,color='navy')
plt2.yticks(y_pos, classifier)
plt2.xlabel('F1 Score')
plt2.title('Classification Performance')
plt2.show()
```



In [ ]: