InceptionV3 Dataslayer

Tristan Khayru Abiyudha
Universitas Gadjah Mada

Salman Faiz Hidayat
Universitas Gadjah Mada

Adam Maulana Haq
Universitas Gadjah Mada

December 2024

1 Code

```
import os
import numpy as np
import pandas as pd
import tensorflow as tf
from tensorflow.keras.applications import InceptionV3
from tensorflow.keras.layers import Dense, GlobalAveragePooling2D, Dropout
from tensorflow.keras.models import Model
from tensorflow.keras.preprocessing.image import load_img, img_to_array
from sklearn.model_selection import train_test_split
from sklearn.metrics import f1_score

# Paths to the train and test directories
train_dir = '/kaggle/input/data-slayer-2-0-machine-learning-competition/train'
test_dir = '/kaggle/input/data-slayer-2-0-machine-learning-competition/test'

# Constants
IMG_SIZE = 224  # Resize all images to 224x224
BATCH_SIZE = 32
EPOCHS = 25  # Increased epochs for better training
LEARNING_RATE = 0.0001  # Reduced learning rate for fine-tuning

# Function to preprocess images
def preprocess_image(image_path):
    try:
        img = load_img(image_path, target_size=(IMG_SIZE, IMG_SIZE))  # Resize
        img = img_to_array(img) / 255.0  # Normalize to [0,1]
        return img
```

```
    except Exception as e:
        print(f"Error loading {image_path}: {e}")
        return None

# Load training data
train_data = []
train_labels = []

for subject in os.listdir(train_dir):
    subject_path = os.path.join(train_dir, subject)
    for category in os.listdir(subject_path):
        category_path = os.path.join(subject_path, category)
        label = 1 if category == 'fall' else 0  # Label fall = 1, non-fall = 0

        for root, _, files in os.walk(category_path):
            for file in files:
                img_path = os.path.join(root, file)
                img = preprocess_image(img_path)
                if img is not None:
                    train_data.append(img)
                    train_labels.append(label)

# Load and preprocess test images
test_images = []
test_ids = []

for img_file in os.listdir(test_dir):
    img_path = os.path.join(test_dir, img_file)
    img = preprocess_image(img_path)
    if img is not None:
        test_images.append(img)
        test_ids.append(img_file)

# Convert to NumPy arrays
train_data = np.array(train_data)
train_labels = np.array(train_labels)
test_images = np.array(test_images)

# Train-validation split
X_train, X_val, y_train, y_val = train_test_split(train_data, train_labels, test_size=0.2, r
print(f"Training samples: {len(X_train)}, Validation samples: {len(X_val)}")

# Load pre-trained InceptionV3 model without the top layer
base_model = InceptionV3(weights='imagenet', include_top=False, input_shape=(IMG_SIZE, IMG_S

# Fine-tuning: Unfreeze some layers
```

```
for layer in base_model.layers[-50:]:
    layer.trainable = True

# Add custom layers on top
x = base_model.output
x = GlobalAveragePooling2D()(x)
x = Dropout(0.3)(x)  # Regularization
x = Dense(128, activation='relu')(x)
x = Dropout(0.3)(x)  # Regularization
output = Dense(1, activation='sigmoid')(x)  # Binary classification

# Create the model
model = Model(inputs=base_model.input, outputs=output)

# Compile the model
model.compile(optimizer=tf.keras.optimizers.Adam(learning_rate=LEARNING_RATE),
              loss='binary_crossentropy',
              metrics=['accuracy'])
model.summary()

# Early stopping callback to prevent overfitting
early_stopping = tf.keras.callbacks.EarlyStopping(monitor='val_loss', patience=5, restore_be

# Train the model
history = model.fit(X_train, y_train,
                    validation_data=(X_val, y_val),
                    epochs=EPOCHS,
                    batch_size=BATCH_SIZE,
                    verbose=1,
                    callbacks=[early_stopping])

# Generate predictions as probabilities on validation set
val_probs = model.predict(X_val)

# Find the best threshold using F1 score
thresholds = np.arange(0.1, 1.0, 0.01)
f1_scores = [f1_score(y_val, (val_probs > t).astype(int)) for t in thresholds]
best_threshold_index = np.argmax(f1_scores)
best_threshold = thresholds[best_threshold_index]
best_f1 = f1_scores[best_threshold_index]

print(f"Best Threshold: {best_threshold}")
print(f"Best F1-Score: {best_f1}")

# Predict labels for test data using the optimized threshold
predictions = model.predict(test_images)
```

```
predicted_labels = (predictions > best_threshold).astype(int).reshape(-1)

# Save predictions to sample_submission.csv
submission = pd.DataFrame({'id': test_ids, 'label': predicted_labels})
submission.to_csv('submission.csv', index=False)
print("Submission file saved!")

# Save the model to an H5 file
model.save('inceptionv3_model.h5')
print("Model saved to inceptionv3_model.h5")

from sklearn.metrics import confusion_matrix, classification_report
import seaborn as sns
import matplotlib.pyplot as plt

# Get validation predictions
y_val_pred = (model.predict(X_val) > 0.5).astype(int)

# Confusion Matrix
cm = confusion_matrix(y_val, y_val_pred)
sns.heatmap(cm, annot=True, fmt="d", cmap="Blues", xticklabels=["Non-Fall", "Fall"], ytickla
plt.title("Confusion Matrix")
plt.xlabel("Predicted Label")
plt.ylabel("True Label")
plt.show()

# Plot accuracy and loss
plt.figure(figsize=(12, 4))

# Accuracy plot
plt.subplot(1, 2, 1)
plt.plot(history.history['accuracy'], label='Train Accuracy')
plt.plot(history.history['val_accuracy'], label='Val Accuracy')
plt.title('Training and Validation Accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()

# Loss plot
plt.subplot(1, 2, 2)
plt.plot(history.history['loss'], label='Train Loss')
plt.plot(history.history['val_loss'], label='Val Loss')
plt.title('Training and Validation Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
```

```
plt.show()

import matplotlib.pyplot as plt
from PIL import Image

for img_id, pred in zip(test_ids[:5], predictions[:5]):  # Show the first 5 images
    img_path = os.path.join(test_dir, img_id)
    img = Image.open(img_path)

    plt.imshow(img)
    plt.title(f"Prediction: {'Fall' if pred >  best_threshold else 'No Fall'}, Probability:
    plt.axis('off')
    plt.show()
```

2 Explanation

Essentially, Inception neural network in general works by expanding the network, or widening, as opposed to extending the layer deeper like ResNet does. It processes the output of a layer into multiple transformations using different convolution or multiple processes (such as pooling) before entering the next layer. The result of each filter or layer will be concatenated before entering the next layer [3]. This addresses the issue of choosing which filter or operation is the most optimized for the image [2].

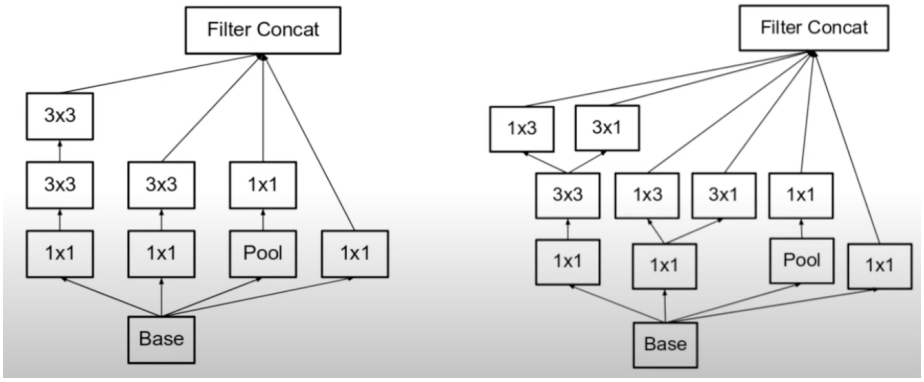

Figure 1: Illustration of Widening layers of InceptionV2

Although there are some additional features in InceptionV3 compared to previous versions. Figure 2 shows the overall architecture of InceptionV3 that is more optimized and reducing complexity:

1. Factorized Convolutions: Instead of using larger convolutions, InceptionV3 employs smaller convolutions, such as factorizing a 5×5 convolution into two successive 3×3 convolutions. This approach reduces computational costs and mitigates overfitting.

2. Auxiliary Classifiers: To address the vanishing gradient problem in deep networks, InceptionV3 includes auxiliary classifiers at intermediate layers. These classifiers provide additional gradient signals during training, enhancing convergence and acting as regularizers.

3. Label Smoothing: InceptionV3 introduces label smoothing during training, which distributes some probability mass from the true label to other labels. This technique helps prevent overfitting and improves model calibration.

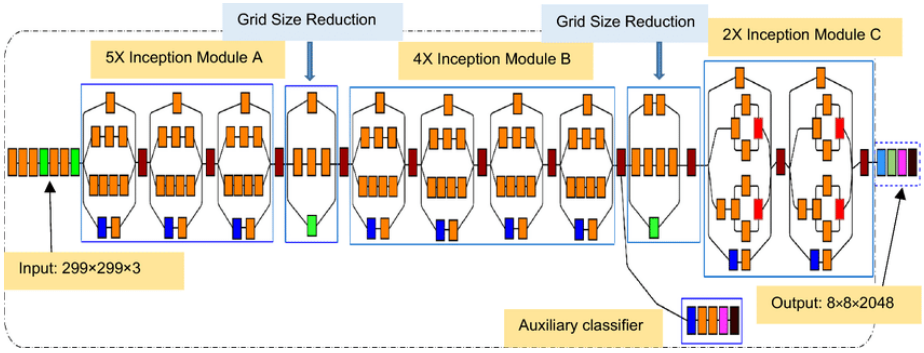

Figure 2: Illustration of InceptionV3 Architecture [1]

References

[1] Block diagram of inception-v3 improved deep architecture — download scientific diagram.

[2] Asrar G. Alharthi and Salha M. Alzahrani. Do it the transformer way: A comprehensive review of brain and vision transformers for autism spectrum disorder diagnosis and classification. *Computers in Biology and Medicine*, 167, 12 2023.

[3] Nicolai Nielsen. (536) inception explained: Understanding the architecture and module of inception networks - youtube, 5 2021.