5.1.1 Πλήρης Λίστα του Κώδικα που χρησιμοποιείται στο Έργο

Κώδικας προ επεξεργασίας Δεδομένων:

Στον παρακάτω κώδικα γίνεται εισαγωγή της εικόνας → αλλαγή του μεγέθους της (256,256) → και κανονικοποίηση

```
import cv2
import numpy as np

# Load image
image = cv2.imread(img_path)
image = cv2.resize(image, self.image_size)
image = image / 255.0 # Normalize to [0, 1]
```

Κώδικας 1: Προ επεξεργασία δεδομένων

Εκπαίδευση Μοντέλου:

Στην εκπαίδευση του μοντέλου περιλαμβάνονται τεχνικές όπως ***ModelCheckPoint*** Και ***EarlyStopping***

```
from tensorflow.keras.callbacks import ModelCheckpoint, EarlyStopping

checkpoint_callback = ModelCheckpoint('/content/drive/My Drive/Colab
Notebooks/model_checkpoint.h5', save_weights_only=True,
save_best_only=True, monitor='val_loss', mode='min', verbose=1)
early_stop = EarlyStopping(patience=3, restore_best_weights=True)

history = atrous_unet.fit(train_gen, validation_data=val_gen,
epochs=14, callbacks=[checkpoint_callback, early_stop])
```

Κώδικας 2: Early Stopping και ModelCkeck Point

Συνάρτηση υπολογισμού Recall – Precision – F1 Score – Accuracy

```
def calculate_metrics_from_cm(cm):
    tp = cm[1, 1]
    fp = cm[0, 1]
    fn = cm[1, 0]
    tn = cm[0, 0]

    # Calculate metrics
    precision = tp / (tp + fp) if (tp + fp) != 0 else 0
    recall = tp / (tp + fn) if (tp + fn) != 0 else 0
    f1 = 2 * (precision * recall) / (precision + recall) if (precision
+ recall) != 0 else 0
    accuracy = (tp + tn) / cm.sum() if cm.sum() != 0 else 0

    return precision, recall, f1, accuracy
```

Κώδικας 3: Συνάρτηση υπολογισμού Recall, Precision, F1 Score, Accuracy

Costume Sequence Class for Image-Mask Pairs:

Η συνάρτηση ***ImageMaskGenerator*** είναι μια προσαρμοσμένη κλάση που επεκτείνει την κλάση ***Sequence***. Υλοποιεί μια γεννήτρια εικόνων και μασκών βοηθάει στην φόρτωση και επεξεργασία τους. Έτσι το μοντέλο να λαμβάνει τα δεδομένα στην σωστή μορφή.

```
class ImageMaskGenerator(Sequence):
    def __init__(self, df, batch_size=32, image_size=(256, 256),
shuffle=True, seed=None):
        self.df = df
        self.batch_size = batch_size
        self.image_size = image_size
        self.shuffle = shuffle
        self.seed = seed
        self.indices = np.arange(len(self.df))
        if self.shuffle:
            np.random.seed(self.seed)
            np.random.shuffle(self.indices)

    def __len__(self):
        return int(np.ceil(len(self.df) / self.batch_size))

    def on_epoch_end(self):
        if self.shuffle: np.random.shuffle(self.indices)

    def __getitem__(self, index):
        start_idx = index * self.batch_size
        batch_indices = self.indices[start_idx:start_idx +
self.batch_size]
        return self.__data_generation(batch_indices)

    def __data_generation(self, batch_indices):
        batch_images = np.empty((len(batch_indices), *self.image_size,
3), dtype=np.float32)
        batch_masks = np.empty((len(batch_indices), *self.image_size,
1), dtype=np.float32)

        for i, idx in enumerate(batch_indices):
            img_path = self.df.iloc[idx]['image']
            mask_path = self.df.iloc[idx]['mask']

            # Load image and mask
            image = cv2.imread(img_path)
            image = cv2.resize(image, self.image_size)
            image = image / 255.0  # Normalize to [0, 1]

            mask = cv2.imread(mask_path, cv2.IMREAD_GRAYSCALE)
            mask = cv2.resize(mask, self.image_size)
            mask = mask[..., np.newaxis]  # Add channel dimension
            mask = mask / 255.0  # Normalize to [0, 1]

            batch_images[i] = image
            batch_masks[i] = mask
        return batch_images, batch_masks
```

Κώδικας 4: Προσαρμοσμένη κλάση ImageMaskGenerator

Κώδικας Αύξησης Δεδομένων:

Ο κώδικας περιλαμβάνεις τεχνικές αύξησης δεδομένων (ImageDataGenerator), φορτώνοντας τις εικόνες σε παρτίδες.

```
from tensorflow.keras.preprocessing.image import ImageDataGenerator

def create_gens(train_df, valid_df, test_df, batch_size):
    img_size = (256, 256)
    channels = 3
    color = 'rgb'
    img_shape = (img_size[0], img_size[1], channels)
    ts_length = len(test_df)
    test_batch_size = max(sorted([ts_length // n for n in range(1,
ts_length + 1) if ts_length % n == 0 and ts_length / n <= 80]))
    test_steps = ts_length // test_batch_size

    def scalar(img):
        return img

    tr_gen = ImageDataGenerator(preprocessing_function=scalar, horizon-
tal_flip=True)
    ts_gen = ImageDataGenerator(preprocessing_function=scalar)

    train_gen = tr_gen.flow_from_dataframe(train_df, x_col='filepaths',
y_col='labels', target_size=img_size, class_mode='categorical',
                                           color_mode=color, shuf-
fle=True, batch_size=batch_size)

    valid_gen = ts_gen.flow_from_dataframe(valid_df, x_col='filepaths',
y_col='labels', target_size=img_size, class_mode='categorical',
                                           color_mode=color, shuf-
fle=True, batch_size=batch_size)

    test_gen = ts_gen.flow_from_dataframe(test_df, x_col='filepaths',
y_col='labels', target_size=img_size, class_mode='categorical',
                                          color_mode=color, shuf-
fle=False, batch_size=test_batch_size)
    return train_gen, valid_gen, test_gen

train_gen, val_gen, test_gen = create_gens(train_df, val_df, test_df,
batch_size=32)
```

Κώδικας 5: ImageDataGenerator - Αύξηση δεδομένων