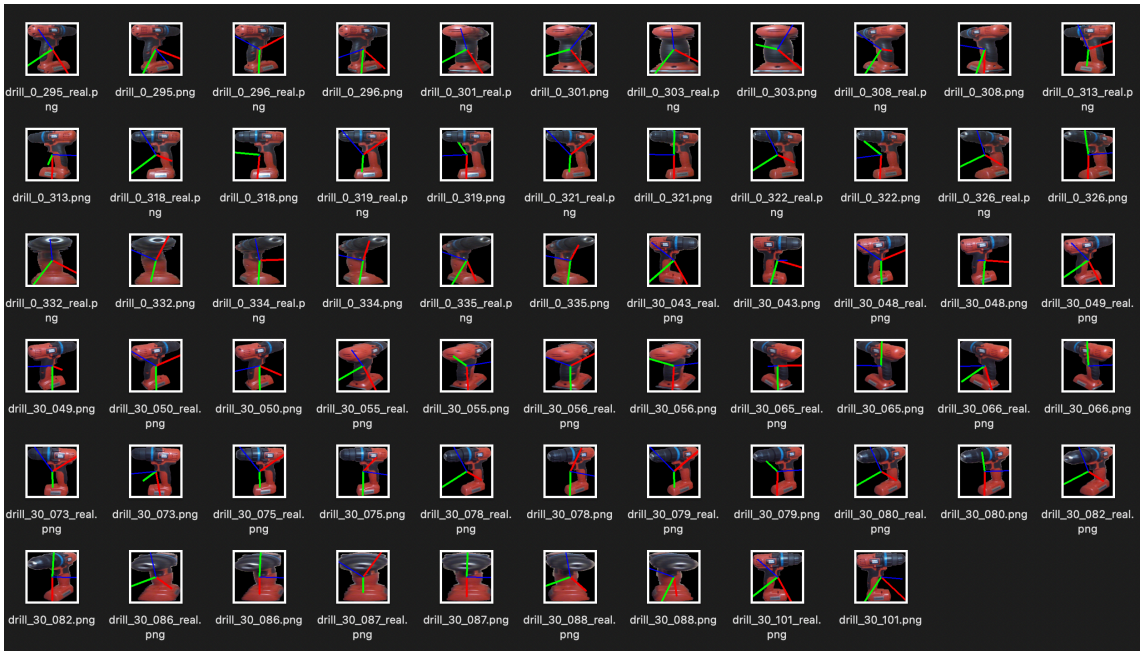
Bartosz Klimza

Sprawozdanie 5 - SqueezeNet

Celem zadania jest zapoznanie z sieciami SqeezeNet.

Na początek przekształcam skrypt z poprzednich zajęć tak, by wykorzystywał sieć SqeezeNet. W tym celu modyfikuję funkcję __create_model(self) w pliku models.py oraz dokonuję kilku poprawek tak, aby działała poprawnie (kod jest załączony razem ze sprawozdaniem).

Sieć trenuję ponownie na zbiorze drill dostępnym wraz z poprzednim zadaniem na 300 epokach. Rezultat:



Podobnie jak przy sieci AlexNet można zauważyć, że linie estymujące pozy na grafikach real oraz z predykcją nieco różnią się od siebie.

Porównanie sieci AlexNet i SqeezeNet

AlexNet:

```
inputs = tf.keras.layers.Input(shape=(self.input_size, self.input_size, 3))

feature = tf.keras.layers.Conv2D(filters=64, kernel_size=(11, 11), strides=4, padding='same', activation=tf.nn.relu)(inputs)
feature = tf.keras.layers.MaxPool2D(pool_size=(3, 3), strides=2)(feature)
feature = tf.keras.layers.Conv2D(filters=192, kernel_size=(5, 5), padding='same', activation=tf.nn.relu)(feature)
feature = tf.keras.layers.MaxPool2D(pool_size=(3, 3), strides=2)(feature)
feature = tf.keras.layers.Conv2D(filters=384, kernel_size=(3, 3), padding='same', activation=tf.nn.relu)(feature)
feature = tf.keras.layers.Conv2D(filters=256, kernel_size=(3, 3), padding='same', activation=tf.nn.relu)(feature)
feature = tf.keras.layers.Conv2D(filters=256, kernel_size=(3, 3), padding='same', activation=tf.nn.relu)(feature)
feature = tf.keras.layers.MaxPool2D(pool_size=(3, 3), strides=2)(feature)
feature = tf.keras.layers.Flatten()(feature)
feature = tf.keras.layers.Dropout(0.5)(feature)
feature = tf.keras.layers.Dense(units=4096, activation=tf.nn.relu)(feature)

fc_yaw = tf.keras.layers.Dense(name='yaw', units=self.class_num)(feature)
fc_pitch = tf.keras.layers.Dense(name='pitch', units=self.class_num)(feature)
fc_roll = tf.keras.layers.Dense(name='roll', units=self.class_num)(feature)


model = tf.keras.Model(inputs=inputs, outputs=[fc_yaw, fc_pitch, fc_roll])
```

SqeezeNet:

```
    inputs = Input(shape=(self.input_size, self.input_size, 3))
    conv1 = Conv2D(96, kernel_size=(3, 3), strides=(2, 2), padding='same', activation='relu', name='Conv1')(inputs)
    maxpool1 = MaxPooling2D(pool_size=(2, 2), strides=(2, 2), name='Maxpool1')(conv1)
    batch1 = BatchNormalization(name='Batch1')(maxpool1)
    fire4 = fire_module(batch1, 32, 128, 128, "Fire2")
    maxpool4 = MaxPooling2D(pool_size=(3, 3), strides=(2, 2), name='Maxpool2')(fire4)
    fire6 = fire_module(maxpool4, 48, 192, 192, "Fire3")
    fire7 = fire_module(fire6, 48, 192, 192, "Fire4")
    fire8 = fire_module(fire7, 48, 192, 192, "Fire5")
    maxpool8 = MaxPooling2D(pool_size=(3, 3), strides=(2, 2), name='Maxpool5')(fire8)
    dropout = Dropout(0.5, name="Dropout")(maxpool8)
    conv10 = Conv2D(10, kernel_size=(1, 1), strides=(1, 1), padding='same', activation='relu', name='Conv6')(dropout)
    batch10 = BatchNormalization(name='Batch6')(conv10)
    avgpool10 = GlobalAveragePooling2D(name='GlobalAvgPool6')(batch10)

    fc_yaw = tf.keras.layers.Dense(name='yaw', units=self.class_num)(avgpool10)
    fc_pitch = tf.keras.layers.Dense(name='pitch', units=self.class_num)(avgpool10)
    fc_roll = tf.keras.layers.Dense(name='roll', units=self.class_num)(avgpool10)

    model = Model(inputs=inputs, outputs=[fc_yaw, fc_pitch, fc_roll])
```

Jak można zauważyć sieć AlexNet posiada większą ilość warstw typu Conv2D i Dense oraz korzysta z warstwy typu Flatten. Sieć Sqeeze posiada warstwy typu BatcNormalization, GlobalAveragePooling2D oraz wykorzystuje funkcję fire_module(). Liczba warstw MaxPooling2D jest taka sama w obu sieciach oraz pojawia się w nich dropout.