Evaluacija rezultata

Posmatramo problem detekcije promene na plućima. Potrebno je prepoznati koja pluća su zdrava (pripadaju klasi „normal“), bakteriološki inficirana (pripadaju klasi „bacteria“), ili imaju virusne posledice (pripadaju klasi „virus“).

Nakon klasifikacije, izdvojene su grupe podataka za treninig i za validaciju. Za validaciju je uzeto 20% od ukupnih podataka, a ostatak (80%) za treninranje modela.

U nastavku su rezultati dobijeni primenom različitih modela kao i rezultati dobijeni testiranjem konkretnih modela.

Model 1:

U ovom modelu korišćen je ugrađeni *vgg16* model, jedan *Flatten* sloj i tri skrivena *Dense* sloja koja imaju 100 neurona. Aktivaciona funkcija je *relu*, a za predikciju je *softmax*.

Optimizer je *Adamx*, *learning_rate* je 10^{-5}, pokušali smo i sa većim vrednostima, ali ova je dala najbolje rezultate.

Za loss funkciju je korišćena *categorical_crossentropy* jer odgovara problemu klasifikacije više klasa.

```
conv_model = tf.keras.applications.vgg16.VGG16(weights='imagenet', include_top=False, input_shape=(180,180,3))

x = tf.keras.layers.Flatten()(conv_model.output)

x = tf.keras.layers.Dense(100, activation='relu')(x)
x = tf.keras.layers.Dense(100, activation='relu')(x)
x = tf.keras.layers.Dense(100, activation='relu')(x)

predictions = tf.keras.layers.Dense(3, activation='softmax')(x)

full_model = tf.keras.models.Model(inputs=conv_model.input, outputs=predictions)

for layer in conv_model.layers:
    layer.trainable = False

full_model.compile(loss='categorical_crossentropy',
    optimizer=tf.keras.optimizers.Adamax(1e-5),
    metrics=['acc'])
```

Slika 1, Implementacija modela 1

Rezultati modela 1:

Nakon 30 epoha rezultati su prikazani na Slika 2 odakle vidimo da je treniranje završeno i ocenjeno sa: loss = 45%, acc = 80%. Validacija je ocenjena sa: val_loss = 53%, val_acc = 76%.

Nakon treniranja modela, prosleđeni su testni podaci i rezultati su sledeći: loss = 58%, acc = 80%. Zaključujemo da je ovaj model dobro istreniran jer su rezultati dobijeni za trening slični rezultatima testiranja.

```
Epoch 23/30
141/141 [==============================] - 31s 218ms/step - loss: 0.4723 - acc: 0.8040 - val_loss: 0.5526 - val_acc: 0.7514
Epoch 24/30
141/141 [==============================] - 31s 217ms/step - loss: 0.4693 - acc: 0.8057 - val_loss: 0.5499 - val_acc: 0.7609
Epoch 25/30
141/141 [==============================] - 31s 217ms/step - loss: 0.4662 - acc: 0.8066 - val_loss: 0.5410 - val_acc: 0.7676
Epoch 26/30
141/141 [==============================] - 31s 217ms/step - loss: 0.4640 - acc: 0.8057 - val_loss: 0.5412 - val_acc: 0.7543
Epoch 27/30
141/141 [==============================] - 31s 217ms/step - loss: 0.4598 - acc: 0.8097 - val_loss: 0.5340 - val_acc: 0.7552
Epoch 28/30
141/141 [==============================] - 30s 216ms/step - loss: 0.4579 - acc: 0.8064 - val_loss: 0.5396 - val_acc: 0.7543
Epoch 29/30
141/141 [==============================] - 31s 217ms/step - loss: 0.4557 - acc: 0.8078 - val_loss: 0.5431 - val_acc: 0.7600
Epoch 30/30
141/141 [==============================] - 31s 217ms/step - loss: 0.4546 - acc: 0.8088 - val_loss: 0.5392 - val_acc: 0.7638
Found 624 images belonging to 3 classes.
21/21 [==============================] - 7s 335ms/step - loss: 0.5868 - acc: 0.8077
Loss of the model is -  58.67677330970764 %
21/21 [==============================] - 7s 321ms/step - loss: 0.5928 - acc: 0.7981
Accuracy of the model is -  79.80769276618958 %
```

Slika 2, Rezultati modela 1

Model 2:

Prethodnom modelu dodali smo jos 3 skrivena *Dense* sloja, i *Flatten* sloj i promenili optimizer funkciju. Predpostavka je bila da će raditi preciznije i dati bolje rezultate.

```
conv_model = tf.keras.applications.vgg16.VGG16(weights='imagenet', include_top=False, input_shape=(180,180,3))
x = tf.keras.layers.Flatten()(conv_model.output)
x = tf.keras.layers.Dense(100, activation='relu')(x)
x = tf.keras.layers.Dense(100, activation='relu')(x)
x = tf.keras.layers.Dense(100, activation='relu')(x)
x = tf.keras.layers.Flatten()(conv_model.output)
x = tf.keras.layers.Dense(100, activation='relu')(x)
x = tf.keras.layers.Dense(100, activation='relu')(x)
x = tf.keras.layers.Dense(100, activation='relu')(x)
predictions = tf.keras.layers.Dense(3, activation='softmax')(x)
full_model = tf.keras.models.Model(inputs=conv_model.input, outputs=predictions)
for layer in conv_model.layers:
    layer.trainable = False
full_model.compile(loss='categorical_crossentropy',
    optimizer=tf.keras.optimizers.SGD(lr=0.01, momentum=0.9),
    metrics=['acc'])
history = full_model.fit_generator(
    training_set,
    validation_data = validation_generator,
    workers=10,
    epochs=10
  )
```

Slika 3, Implementacija modela 2

Rezultati za model 2:

Rezultati za model 2 se ne razlikuju u velikoj meri od početnog modela. Nismo dobili poboljšanje koje smo očekivali.

```
Epoch 3/10
141/141 [==============================] - 38s 272ms/step - loss: 0.5959 - acc: 0.7254 - val_loss: 0.6586 - val_acc: 0.7131
Epoch 4/10
141/141 [==============================] - 38s 272ms/step - loss: 0.5109 - acc: 0.7760 - val_loss: 0.5194 - val_acc: 0.7633
Epoch 5/10
141/141 [==============================] - 39s 273ms/step - loss: 0.5135 - acc: 0.7883 - val_loss: 0.5281 - val_acc: 0.7424
Epoch 6/10
141/141 [==============================] - 38s 272ms/step - loss: 0.4996 - acc: 0.7805 - val_loss: 0.5378 - val_acc: 0.7348
Epoch 7/10
141/141 [==============================] - 38s 273ms/step - loss: 0.4909 - acc: 0.7869 - val_loss: 0.5394 - val_acc: 0.7415
Epoch 8/10
141/141 [==============================] - 39s 274ms/step - loss: 0.4776 - acc: 0.7878 - val_loss: 0.5371 - val_acc: 0.7633
Epoch 9/10
141/141 [==============================] - 39s 275ms/step - loss: 0.4765 - acc: 0.7902 - val_loss: 0.5049 - val_acc: 0.7708
Epoch 10/10
141/141 [==============================] - 39s 274ms/step - loss: 0.4721 - acc: 0.7907 - val_loss: 0.5294 - val_acc: 0.7699
Found 624 images belonging to 3 classes.
21/21 [==============================] - 7s 339ms/step - loss: 0.7204 - acc: 0.7885
Loss of the model is -  72.0402717590332 %
21/21 [==============================] - 7s 342ms/step - loss: 0.7112 - acc: 0.8013
Accuracy of the model is -  80.12820482254028 %
```

Slika 4, Rezultati modela 2

Model 3:

Koristili smo *Keras Sequential* model, kojem smo dodali pet *Con2d* sloja i isto toliko *MaxPool2D* sloja. Na kraju dodat *Flatten* sloj i dva *Dense* sloja, od kojih jedan koristi *softmax* aktivacionu funkciju.

```
model = tf.keras.models.Sequential()

model.add(Conv2D(32 , (3,3) , strides = 1 , padding = 'same' , activation = 'relu' , input_shape = (150,150,3)))
model.add(MaxPool2D((2,2) , strides = 2 , padding = 'same'))

model.add(Conv2D(64 , (3,3) , strides = 1 , padding = 'same' , activation = 'relu'))
model.add(MaxPool2D((2,2) , strides = 2 , padding = 'same'))

model.add(Conv2D(64 , (3,3) , strides = 1 , padding = 'same' , activation = 'relu'))
model.add(MaxPool2D((2,2) , strides = 2 , padding = 'same'))

model.add(Conv2D(128 , (3,3) , strides = 1 , padding = 'same' , activation = 'relu'))
model.add(MaxPool2D((2,2) , strides = 2 , padding = 'same'))

model.add(Conv2D(256 , (3,3) , strides = 1 , padding = 'same' , activation = 'relu'))
model.add(MaxPool2D((2,2) , strides = 2 , padding = 'same'))

model.add(Flatten())
model.add(Dense(units = 128 , activation = 'relu'))
model.add(Dropout(0.9))
model.add(Dense(units = 3 , activation = 'softmax'))

model.compile(optimizer = 'adam' , loss = 'categorical_crossentropy' , metrics = ['accuracy'])
model.summary()

model.fit(train, epochs=30,  validation_data=val)
```

Slika 5, Implementacija modela 3

Rezultati za model 3:

Rezultati dobijeni nakon treniranja nisu zadovoljavajući. Ovaj model ne pokazuje zavidne performanse iz razloga što je preciznost treniranja, validacije i testiranja oko 70%, međutim dobra strana je što nemamo overfitting.

```
Epoch 23/30
133/133 [==============================] - 28s 212ms/step - loss: 0.6163 - accuracy: 0.7143 - val_loss: 0.5909 - val_accuracy: 0.7178
Epoch 24/30
133/133 [==============================] - 28s 211ms/step - loss: 0.6095 - accuracy: 0.7185 - val_loss: 0.6210 - val_accuracy: 0.6951
Epoch 25/30
133/133 [==============================] - 28s 211ms/step - loss: 0.6121 - accuracy: 0.7193 - val_loss: 0.6074 - val_accuracy: 0.7064
Epoch 26/30
133/133 [==============================] - 28s 212ms/step - loss: 0.6135 - accuracy: 0.7190 - val_loss: 0.6018 - val_accuracy: 0.7121
Epoch 27/30
133/133 [==============================] - 28s 211ms/step - loss: 0.6162 - accuracy: 0.7145 - val_loss: 0.5819 - val_accuracy: 0.7017
Epoch 28/30
133/133 [==============================] - 28s 212ms/step - loss: 0.6158 - accuracy: 0.7169 - val_loss: 0.6146 - val_accuracy: 0.7027
Epoch 29/30
133/133 [==============================] - 28s 212ms/step - loss: 0.6074 - accuracy: 0.7249 - val_loss: 0.5927 - val_accuracy: 0.7216
Epoch 30/30
133/133 [==============================] - 28s 210ms/step - loss: 0.5960 - accuracy: 0.7200 - val_loss: 0.5773 - val_accuracy: 0.7216
Found 624 images belonging to 3 classes.
21/21 [==============================] - 7s 334ms/step - loss: 0.8248 - accuracy: 0.6859
Loss of the model is -  82.47990608215332 %
21/21 [==============================] - 7s 349ms/step - loss: 0.8180 - accuracy: 0.6843
Accuracy of the model is -  68.42948794364929 %
```

Slika 6, Rezultati modela 3

Model 4:

U ovom modelu primenili smo implementaciju datu na zvaničnom sajtu keras.io.. U njihovom primeru treniran je model sa dve klase, što je trebalo prilagoditi na naš problem sa prepoznavanjem tri klase.

```
def make_model(input_shape, num_classes):
    inputs = keras.Input(shape=input_shape)
    x = data_augmentation(inputs)
    x = layers.experimental.preprocessing.Rescaling(1.0 / 255)(x)
    x = layers.Conv2D(32, 3, strides=2, padding="same")(x)
    x = layers.BatchNormalization()(x)
    x = layers.Activation("relu")(x)
    x = layers.Conv2D(64, 3, padding="same")(x)
    x = layers.BatchNormalization()(x)
    x = layers.Activation("relu")(x)

    previous_block_activation = x

    for size in [128, 256, 512, 728]:
        x = layers.Activation("relu")(x)
        x = layers.SeparableConv2D(size, 3, padding="same")(x)
        x = layers.BatchNormalization()(x)

        x = layers.Activation("relu")(x)
        x = layers.SeparableConv2D(size, 3, padding="same")(x)
        x = layers.BatchNormalization()(x)

        x = layers.MaxPooling2D(3, strides=2, padding="same")(x)

        residual = layers.Conv2D(size, 1, strides=2, padding="same")(
            previous_block_activation
        )
        x = layers.add([x, residual])
        previous_block_activation = x

    x = layers.SeparableConv2D(1024, 3, padding="same")(x)
    x = layers.BatchNormalization()(x)
    x = layers.Activation("relu")(x)

    x = layers.GlobalAveragePooling2D()(x)
    x = layers.Dropout(0.5)(x)
    outputs = layers.Dense(units=num_classes, activation="softmax")(x)
    return keras.Model(inputs, outputs)
```

Slika 7, Implementacija modela 4

```
model.compile(
    optimizer = keras.optimizers.Adam(1e-3),
    loss = "sparse_categorical_crossentropy",
    metrics = ["accuracy"],
)
```

Rezultati za model 4:

Zbog velikog overfittinga ovaj model nije dobro istreniran, i nije adekvatan za naš problem.

```
Epoch 44/50
133/133 [==============================] - 25s 189ms/step - loss: 0.4038 - accuracy: 0.8259 - val_loss: 0.4631 - val_accuracy: 0.7936
Epoch 45/50
133/133 [==============================] - 25s 189ms/step - loss: 0.3922 - accuracy: 0.8309 - val_loss: 0.4465 - val_accuracy: 0.7907
Epoch 46/50
133/133 [==============================] - 25s 189ms/step - loss: 0.3848 - accuracy: 0.8254 - val_loss: 0.4570 - val_accuracy: 0.8172
Epoch 47/50
133/133 [==============================] - 25s 189ms/step - loss: 0.3871 - accuracy: 0.8288 - val_loss: 0.8065 - val_accuracy: 0.7282
Epoch 48/50
133/133 [==============================] - 25s 189ms/step - loss: 0.3885 - accuracy: 0.8259 - val_loss: 0.4968 - val_accuracy: 0.7898
Epoch 49/50
133/133 [==============================] - 25s 189ms/step - loss: 0.4042 - accuracy: 0.8247 - val_loss: 0.7234 - val_accuracy: 0.7497
Epoch 50/50
133/133 [==============================] - 25s 189ms/step - loss: 0.3880 - accuracy: 0.8295 - val_loss: 0.8010 - val_accuracy: 0.7197
[8.2131779e-01 4.7615856e-10 1.7868225e-01]
This image is 82.13 percent virus, 17.87 percent bacteria and  0.00 percent normal.
Found 624 files belonging to 3 classes.
20/20 [==============================] - 2s 105ms/step - loss: 6.0772 - accuracy: 0.4696
Loss of the model is -  607.7218055725098 %
20/20 [==============================] - 2s 107ms/step - loss: 6.0772 - accuracy: 0.4696
Accuracy of the model is -  56.955129504203796 %
```

Slika 8, Rezultati za model 4