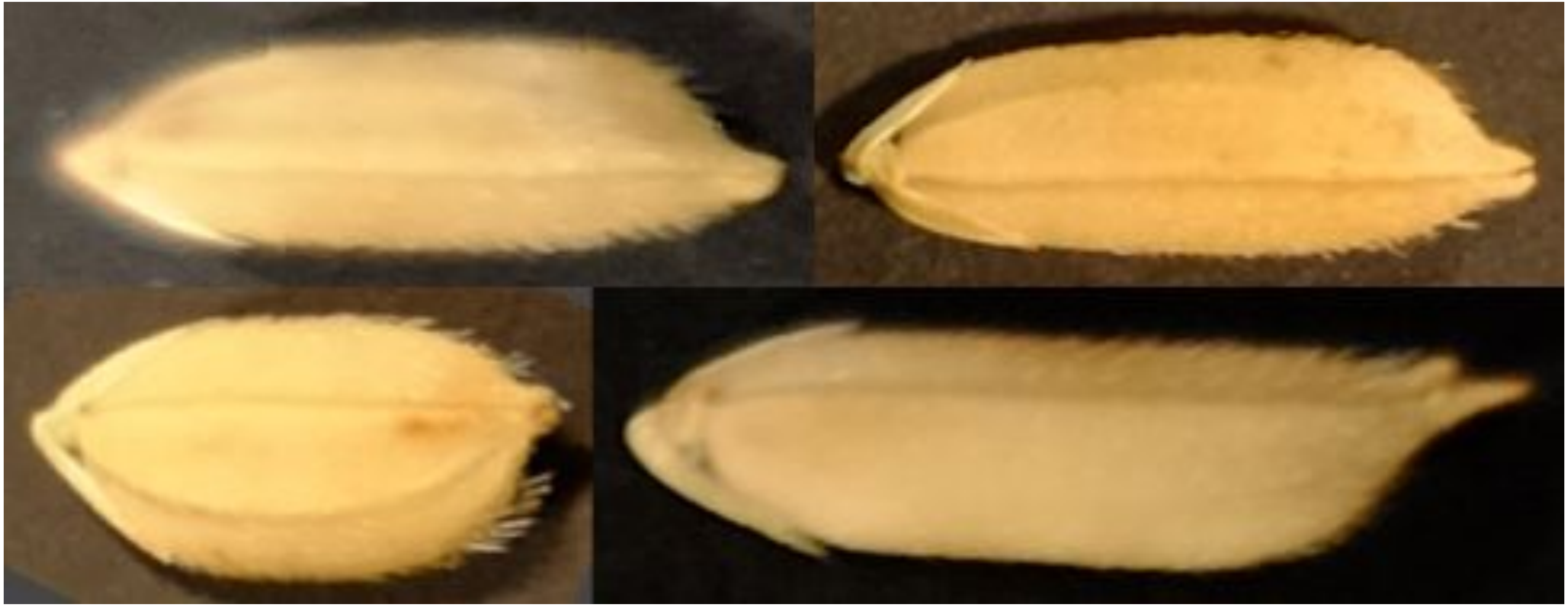
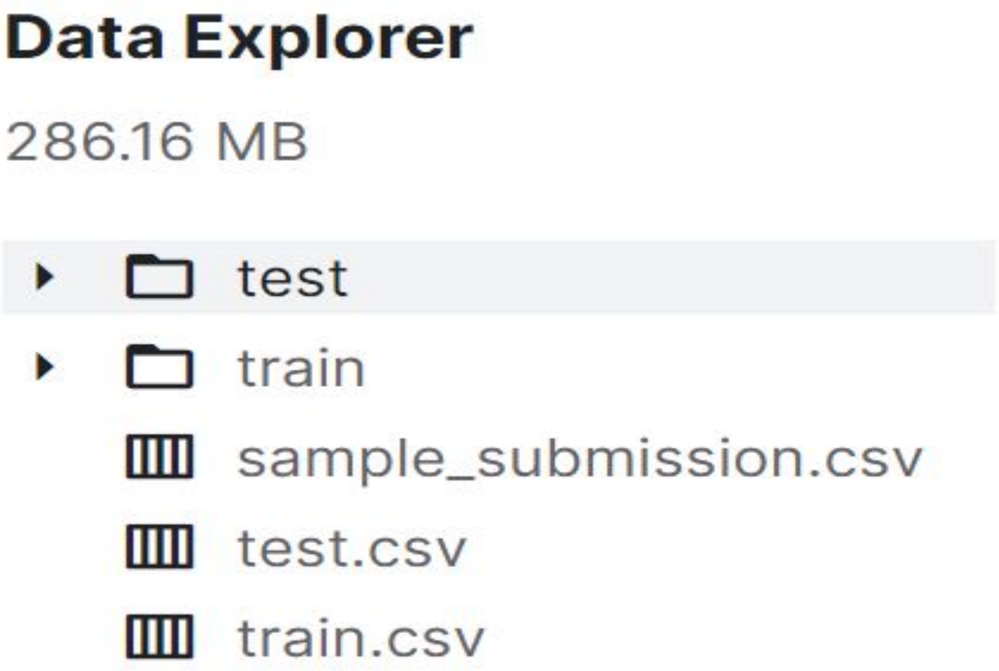
Klasifikacija zrna pirinca

Ilija Doknić

Primer podataka



Struktura foldera

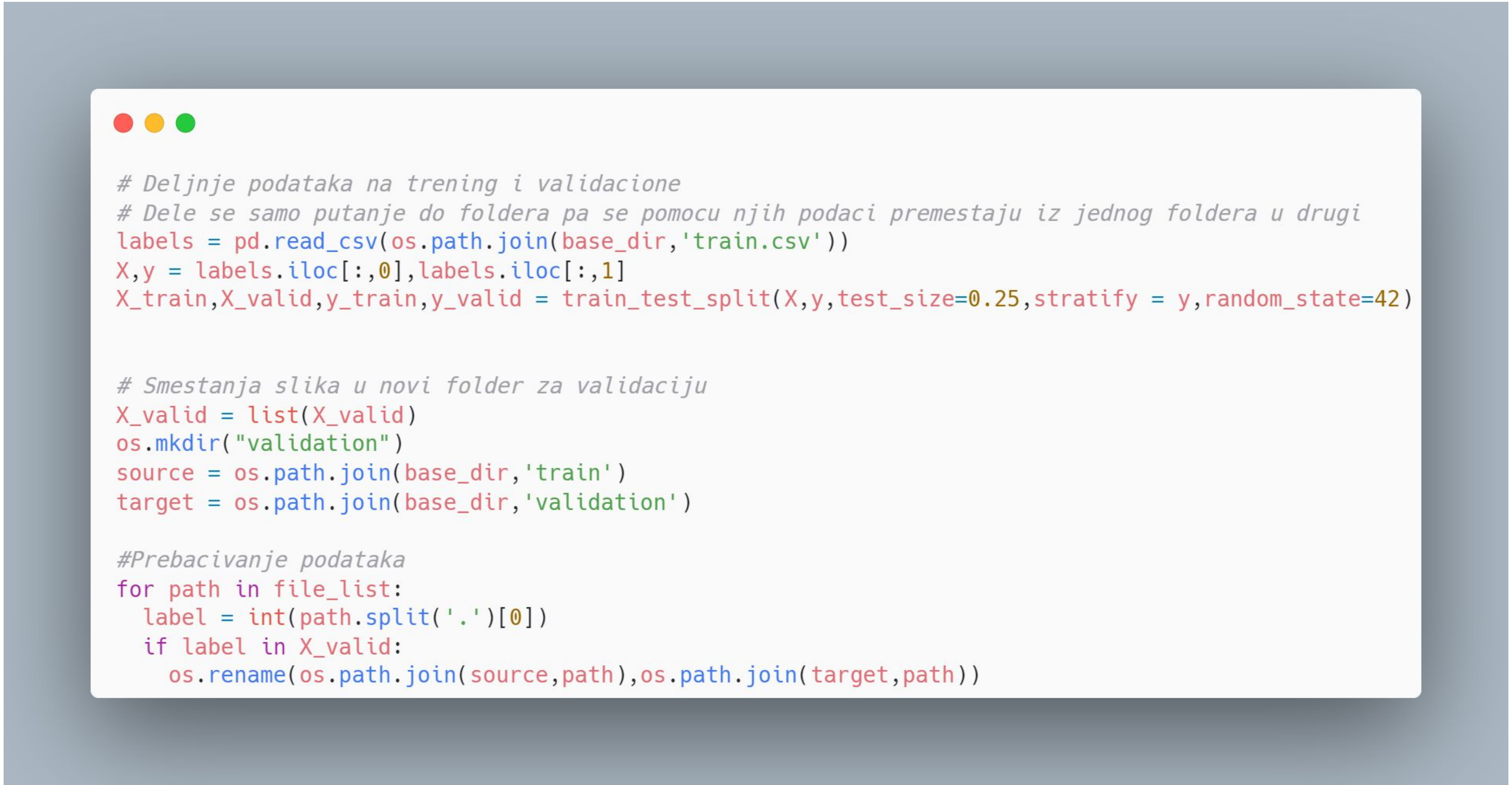


Data Explorer

286.16 MB

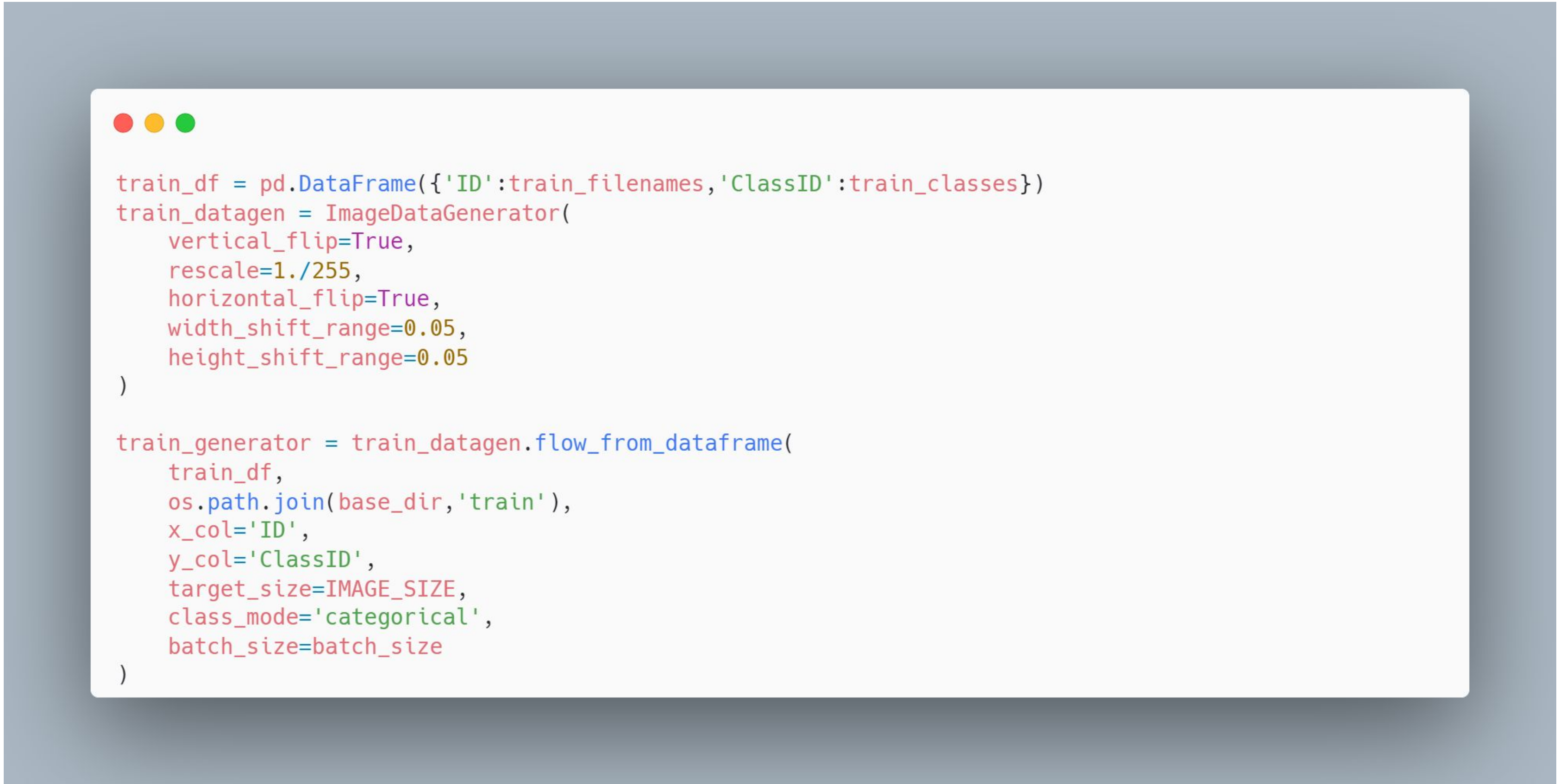
- ▸ test
- ▸ train
- sample_submission.csv
- test.csv
- train.csv

Podela podataka na trening i validacione



```python
# Deljnje podataka na trening i validacione
# Dele se samo putanje do foldera pa se pomocu njih podaci premestaju iz jednog foldera u drugi
labels = pd.read_csv(os.path.join(base_dir,'train.csv'))
X,y = labels.iloc[:,0],labels.iloc[:,1]
X_train,X_valid,y_train,y_valid = train_test_split(X,y,test_size=0.25,stratify = y,random_state=42)


# Smestanja slika u novi folder za validaciju
X_valid = list(X_valid)
os.mkdir("validation")
source = os.path.join(base_dir,'train')
target = os.path.join(base_dir,'validation')

#Prebacivanje podataka
for path in file_list:
  label = int(path.split('.')[0])
  if label in X_valid:
    os.rename(os.path.join(source,path),os.path.join(target,path))
```

Kreiranje generator i augmentacija podataka



```
train_df = pd.DataFrame({'ID':train_filenames,'ClassID':train_classes})
train_datagen = ImageDataGenerator(
    vertical_flip=True,
    rescale=1./255,
    horizontal_flip=True,
    width_shift_range=0.05,
    height_shift_range=0.05
)

train_generator = train_datagen.flow_from_dataframe(
    train_df,
    os.path.join(base_dir,'train'),
    x_col='ID',
    y_col='ClassID',
    target_size=IMAGE_SIZE,
    class_mode='categorical',
    batch_size=batch_size
)
```

Primer augmentovanih slika

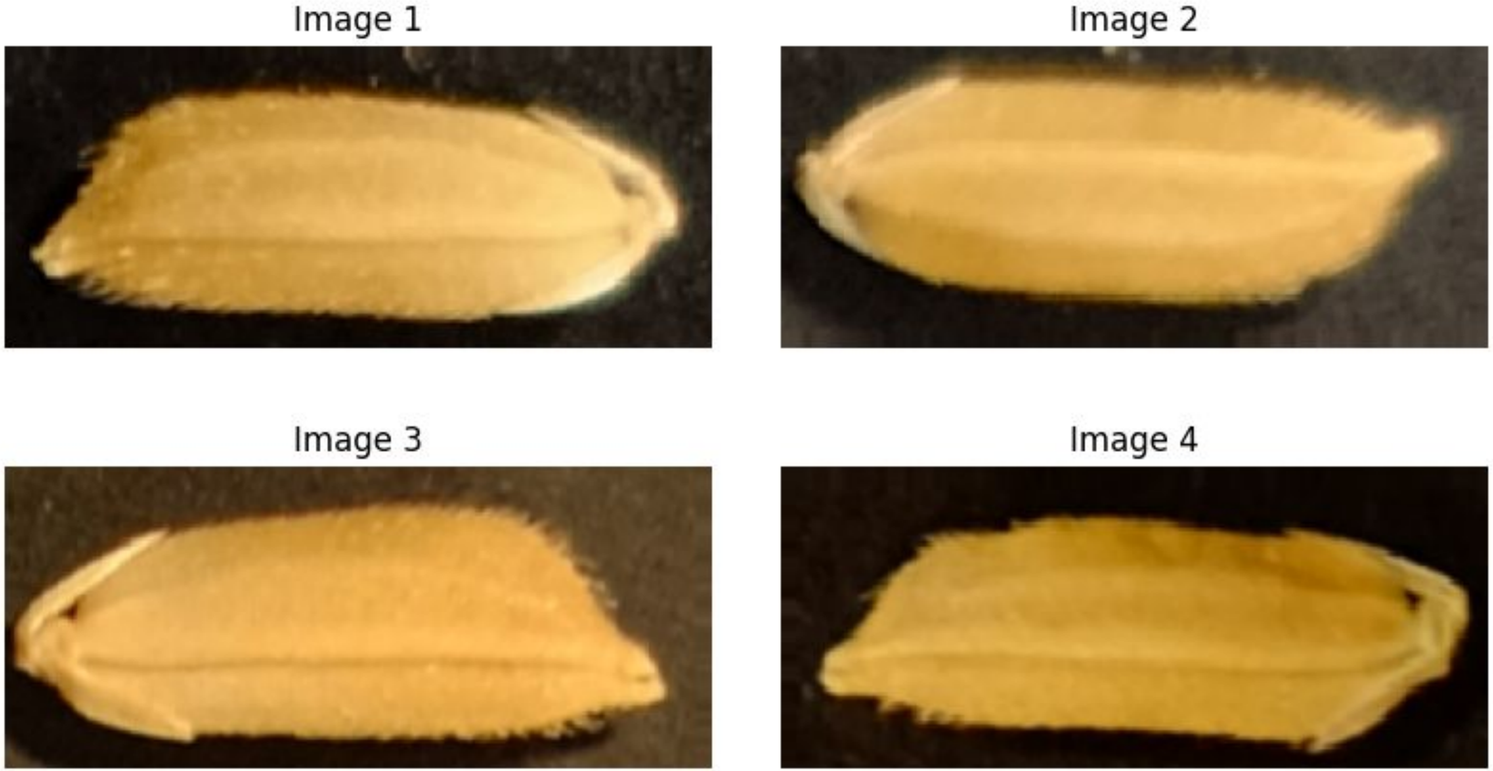


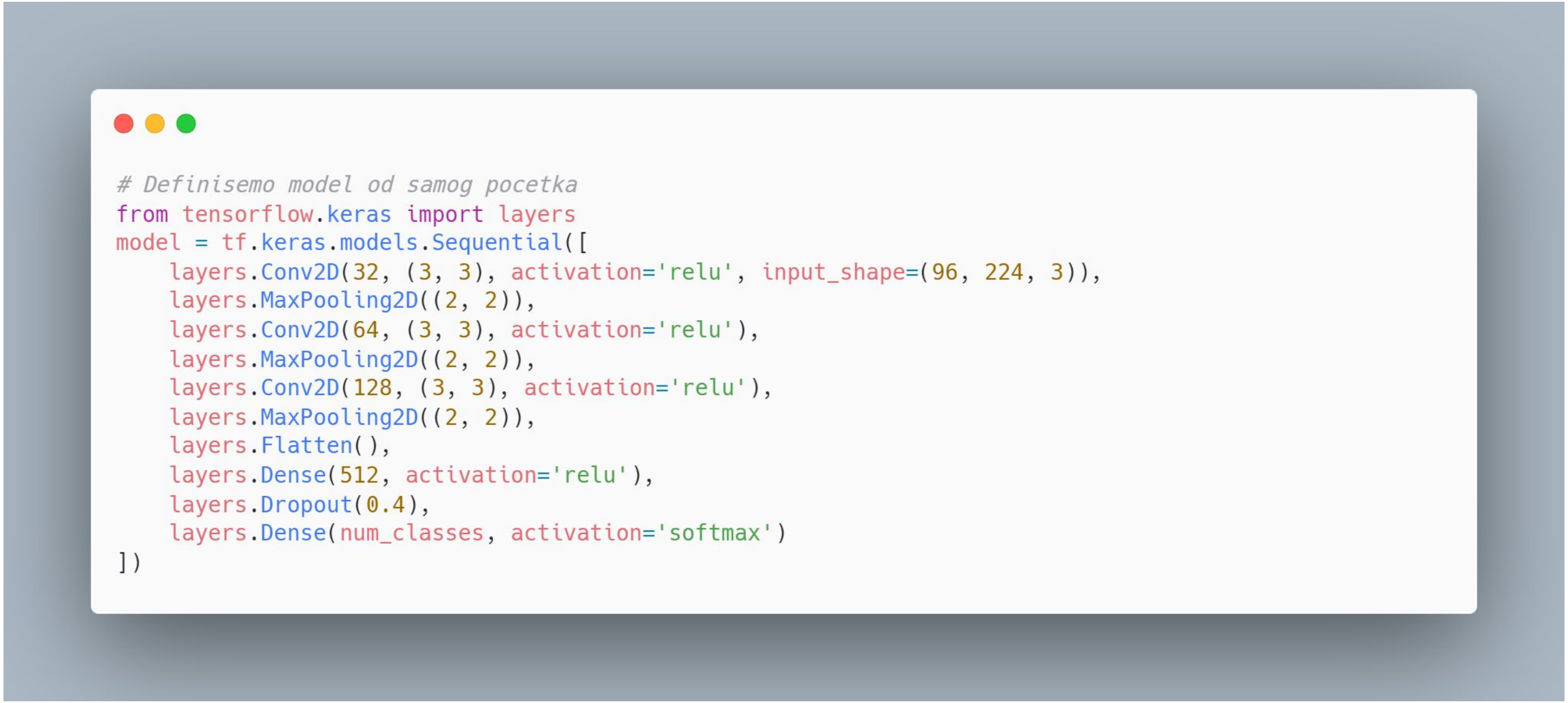
Image 1

Image 2

Image 3

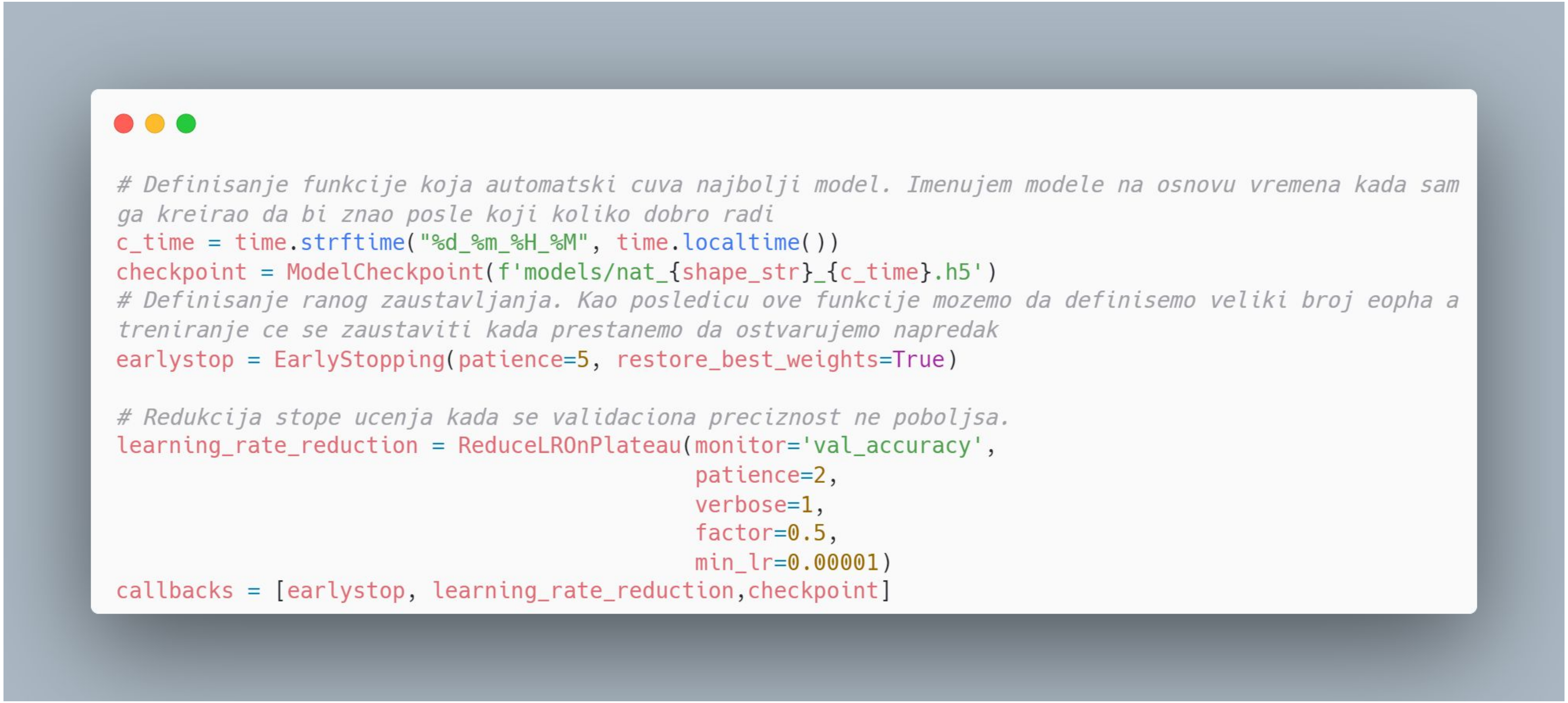
Image 4

Osnovno definisanje modela



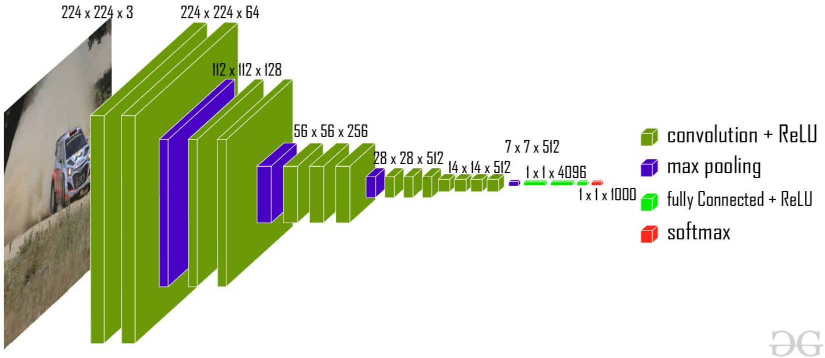
```python
# Definisemo model od samog pocetka
from tensorflow.keras import layers
model = tf.keras.models.Sequential([
    layers.Conv2D(32, (3, 3), activation='relu', input_shape=(96, 224, 3)),
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(64, (3, 3), activation='relu'),
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(128, (3, 3), activation='relu'),
    layers.MaxPooling2D((2, 2)),
    layers.Flatten(),
    layers.Dense(512, activation='relu'),
    layers.Dropout(0.4),
    layers.Dense(num_classes, activation='softmax')
])
```

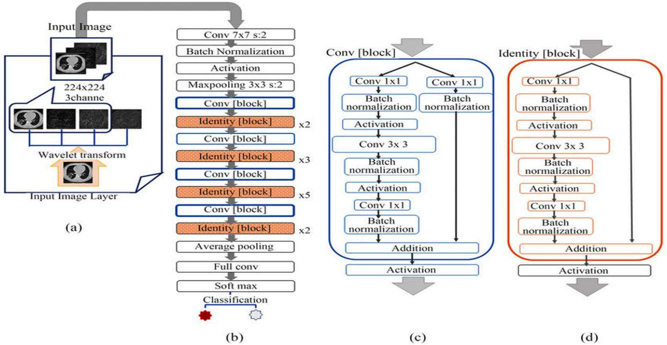
Definisanje callback-ova

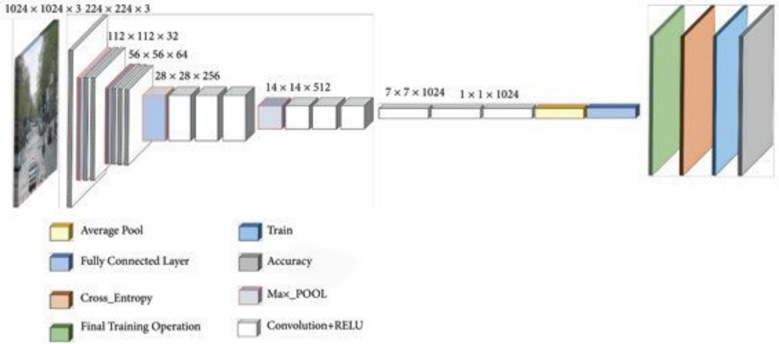


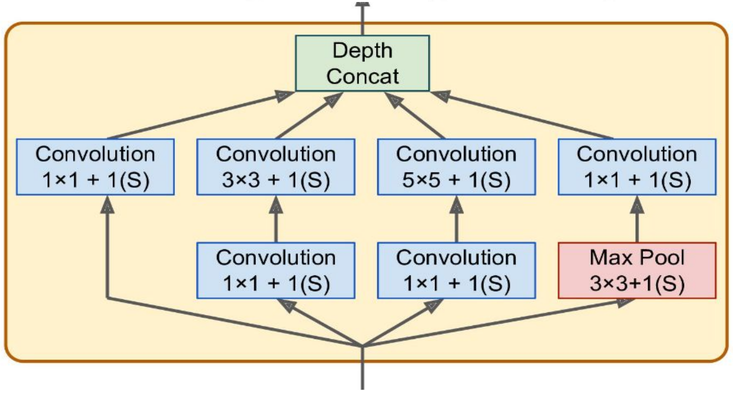
```
# Definisanje funkcije koja automatski cuva najbolji model. Imenujem modele na osnovu vremena kada sam
ga kreirao da bi znao posle koji koliko dobro radi
c_time = time.strftime("%d_%m_%H_%M", time.localtime())
checkpoint = ModelCheckpoint(f'models/nat_{shape_str}_{c_time}.h5')
# Definisanje ranog zaustavljanja. Kao posledicu ove funkcije mozemo da definisemo veliki broj eopha a
treniranje ce se zaustaviti kada prestanemo da ostvarujemo napredak
earlystop = EarlyStopping(patience=5, restore_best_weights=True)

# Redukcija stope ucenja kada se validaciona preciznost ne poboljsa.
learning_rate_reduction = ReduceLROnPlateau(monitor='val_accuracy',
                                            patience=2,
                                            verbose=1,
                                            factor=0.5,
                                            min_lr=0.00001)
callbacks = [earlystop, learning_rate_reduction,checkpoint]
```

Pretrained Neural Networks



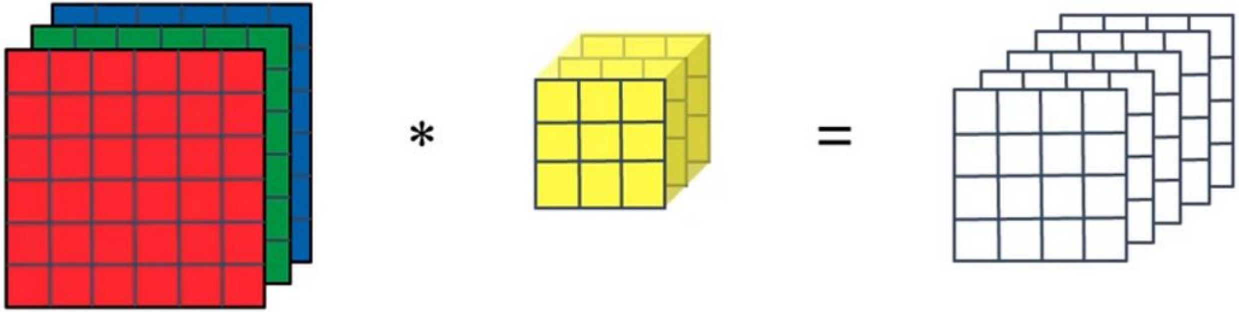




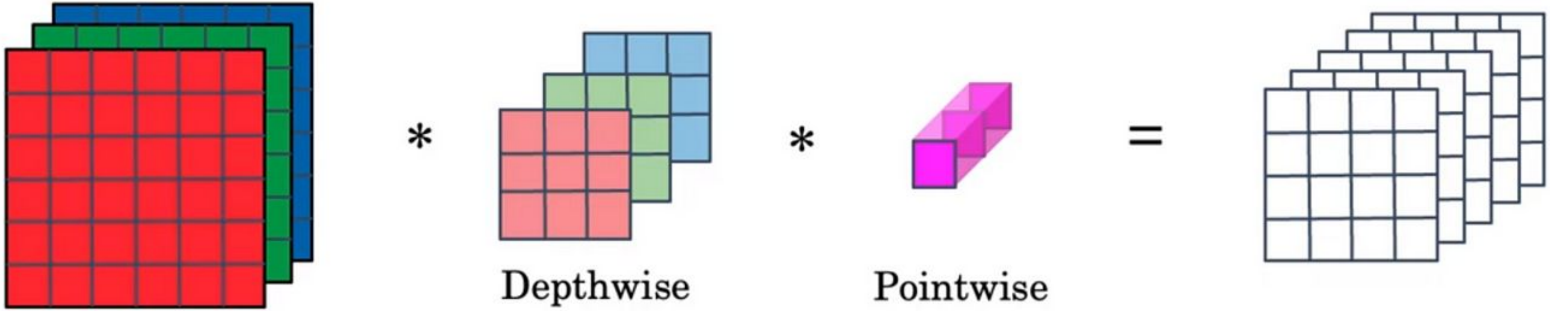


MobileNet

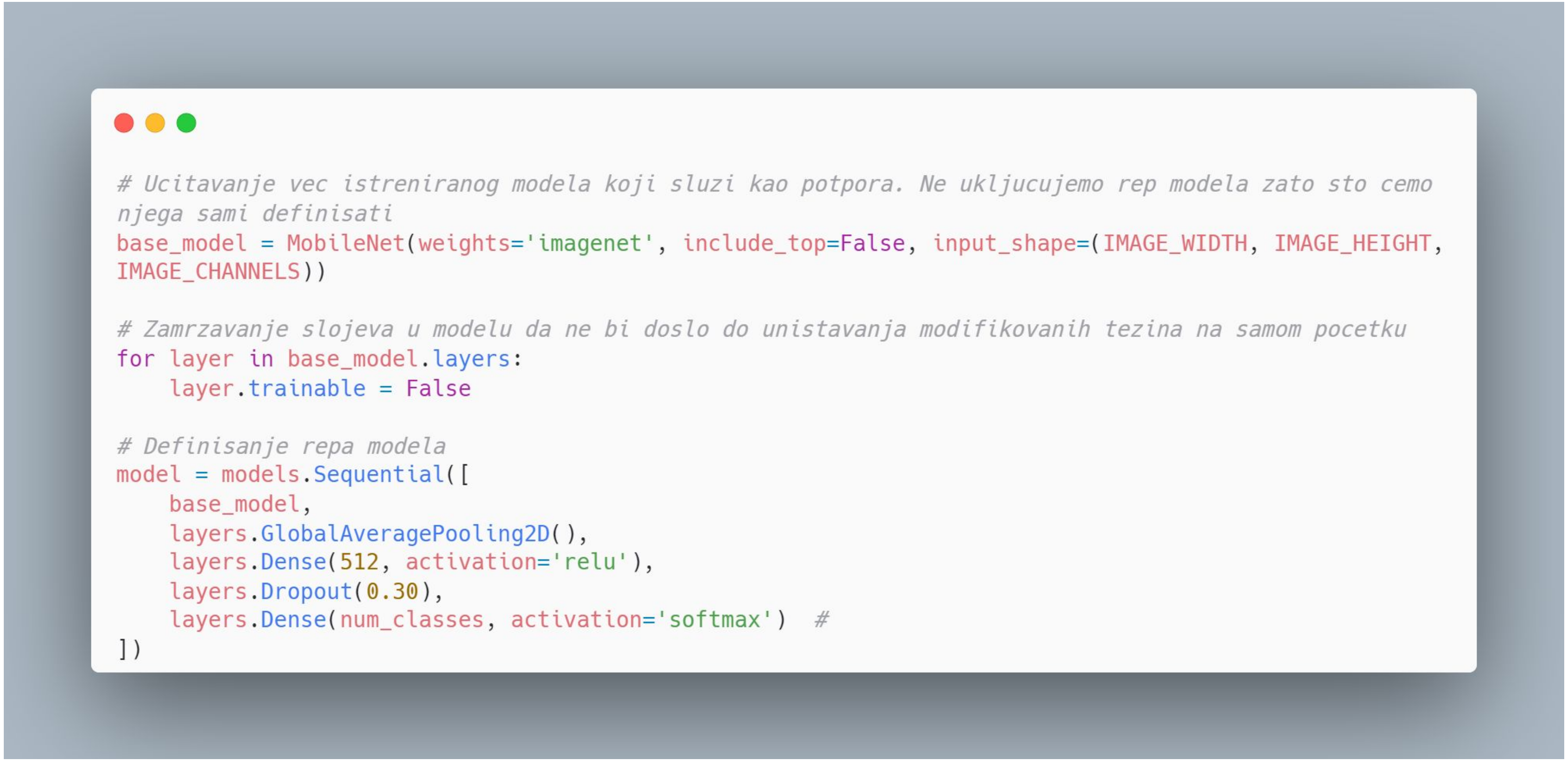
Normal Convolution



Depthwise Separable Convolution

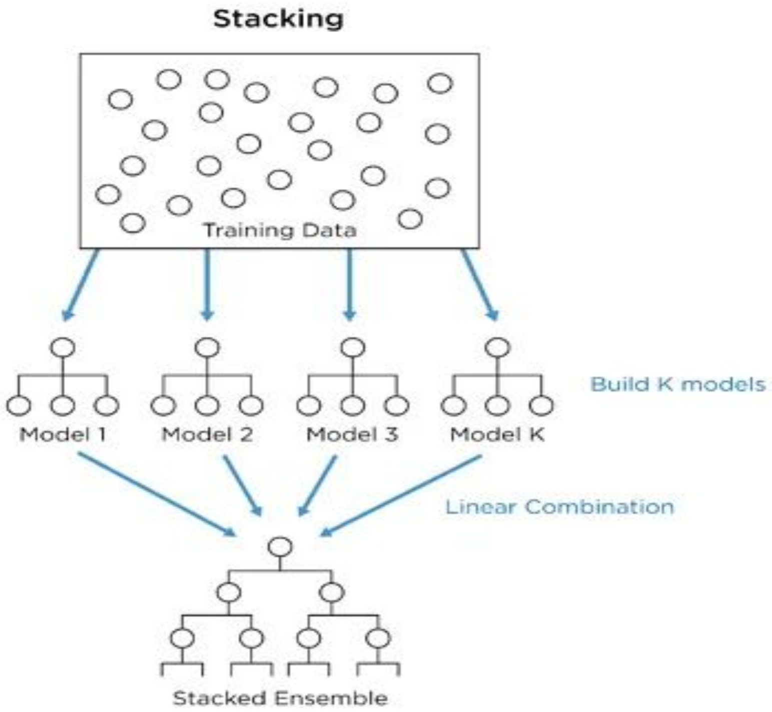


MobileNet

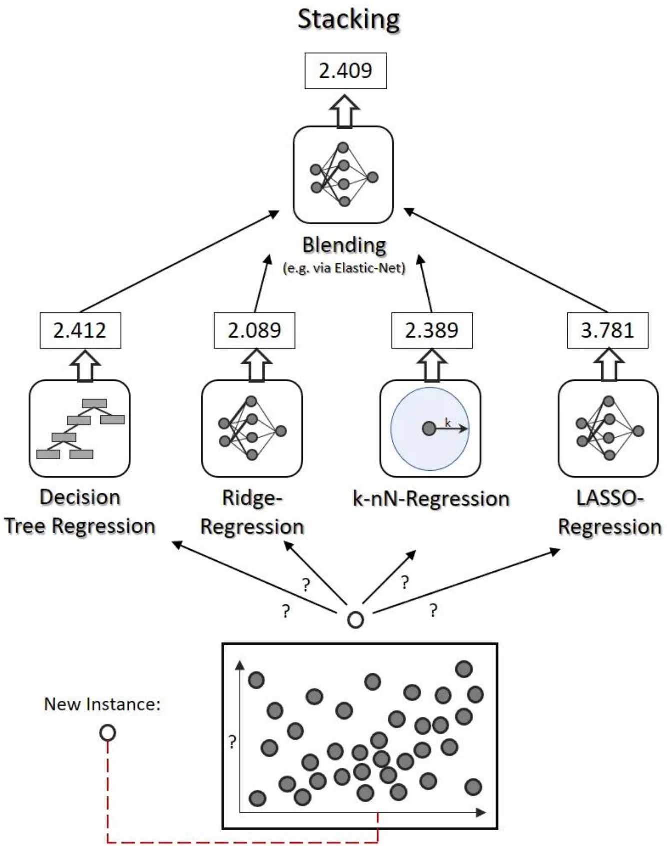


```python
# Ucitavanje vec istreniranog modela koji sluzi kao potpora. Ne ukljucujemo rep modela zato sto cemo njega sami definisati
base_model = MobileNet(weights='imagenet', include_top=False, input_shape=(IMAGE_WIDTH, IMAGE_HEIGHT, IMAGE_CHANNELS))

# Zamrzavanje slojeva u modelu da ne bi doslo do unistavanja modifikovanih tezina na samom pocetku
for layer in base_model.layers:
    layer.trainable = False

# Definisanje repa modela
model = models.Sequential([
    base_model,
    layers.GlobalAveragePooling2D(),
    layers.Dense(512, activation='relu'),
    layers.Dropout(0.30),
    layers.Dense(num_classes, activation='softmax')  #
])
```

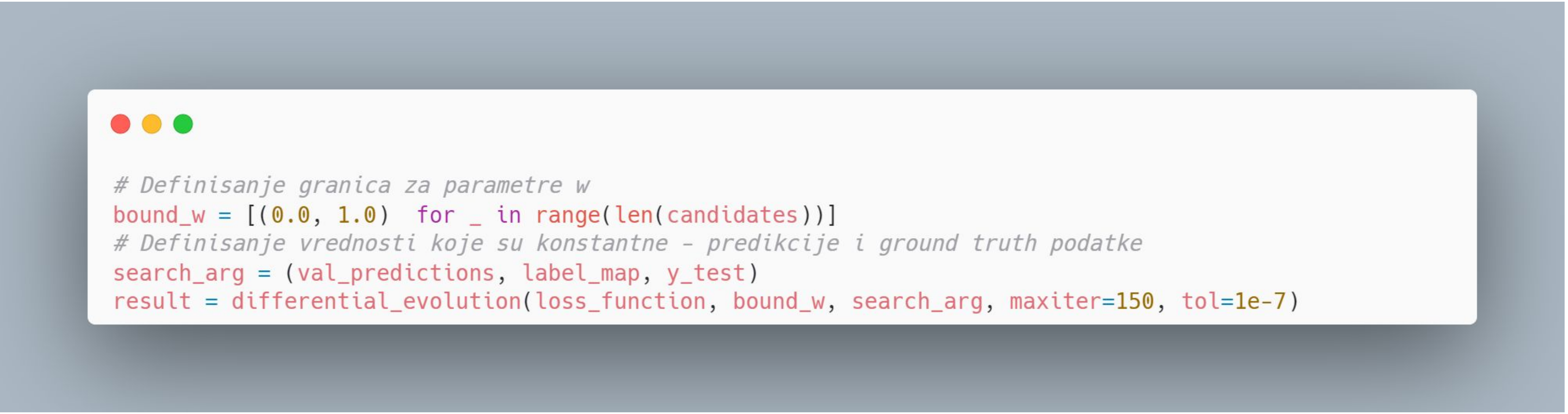
Ensemble learning



Ensemble learning - Stacking



Evolutionary ensemble



```python
# Definisanje granica za parametre w
bound_w = [(0.0, 1.0)  for _ in range(len(candidates))]
# Definisanje vrednosti koje su konstantne - predikcije i ground truth podatke
search_arg = (val_predictions, label_map, y_test)
result = differential_evolution(loss_function, bound_w, search_arg, maxiter=150, tol=1e-7)
```

Hvala na paznji!