Epksperimenti

Friday, July 6, 2018 10:41 AM

Podaci su podeljeni u tri grupe:

- Train ~28000
- Validation ~4000
- Test ~4000

Data:

0: -4593 images- *Angry*
1: -547 images- *Disgust*
2: -5121 images- *Fear*
3: -8989 images- *Happy*
4: -6077 images- *Sad*
5: -4002 images- *Surprise*
6: -6198 images- *Neutral*

Primerak podataka:

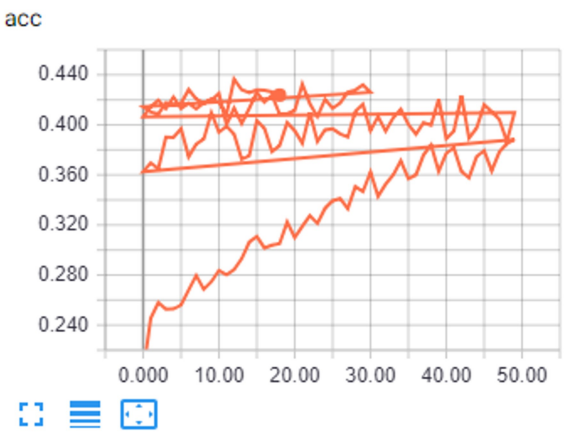
1.
Conv2D(64, 3, relu, l2(0.01))

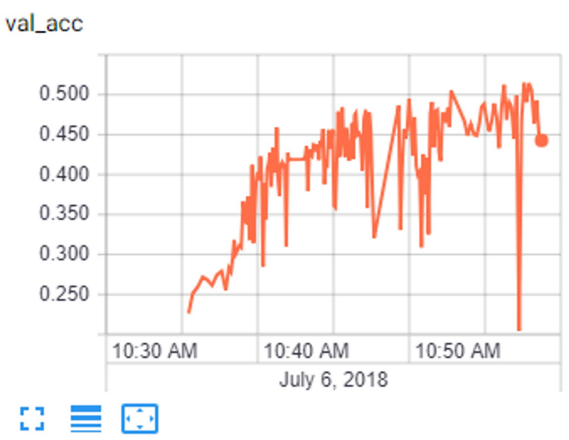
2x Conv2D(128, 3)
BatchNorm()
Relu()
MaxPool(2, 2)

2x Conv2D(512, 3, relu)
BatchNorm()
Relu()
MaxPool(2, 2)
Dropout(0.5)

Dense(512)
Dropout(0.5)
Relu()

Dense(7, softmax)

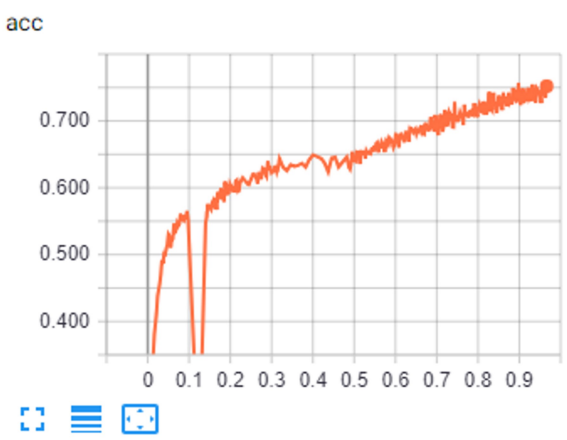


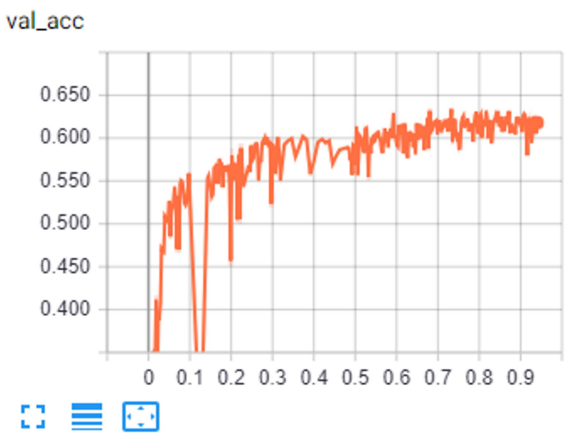


2.
Conv2D(64, 3, relu)

2x Conv2D(128, 3)
BatchNorm()
Relu()
MaxPool(2, 2)

Conv2D(512, 3, relu)
BatchNorm()
Relu()
MaxPool(2, 2)





Conv2D(512, 3, relu, l2(0.001))
BatchNorm()
Relu()
MaxPool(2, 2)

Dense(512)
Relu()

Dense(7, softmax)

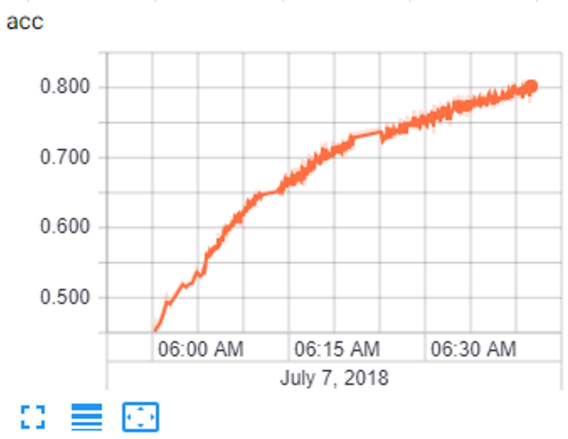
Adagrad
Adadelta

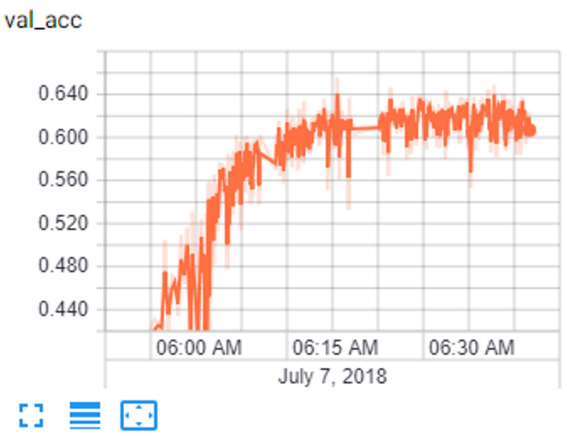
SGD Default
loss: 1.3323 - acc: 0.6309 - val_loss: 1.4286 - val_acc: 0.6016

Adadelta isto samo malo brze

Adagrad zasad sporije, ali mozda ima nade da ode dalje, psticemo jos 100

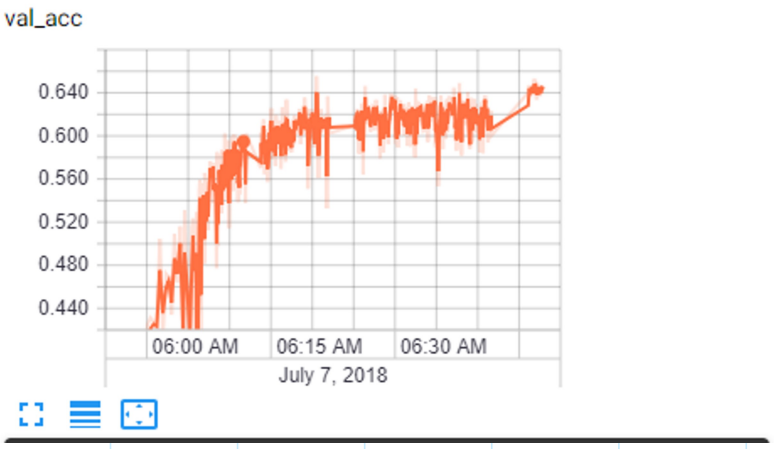
```
model = Sequential()
model.add(Conv2D(
input_shape=(48, 48, 1),
filters=64,
kernel_size=(3, 3),
kernel_initializer=VarianceScaling(),
activation='relu',
trainable=False
))
model.add(MaxPooling2D(
pool_size=(2, 2),
strides=(2, 2)
))
model.add(Conv2D(
filters=128,
kernel_size=(3, 3),
kernel_initializer=VarianceScaling(),
use_bias=False,
trainable=False
))
model.add(BatchNormalization(axis=-1))
model.add(Activation('relu'))
model.add(MaxPooling2D(
pool_size=(2, 2),
strides=(2, 2)
))
model.add(Conv2D(
filters=512,
kernel_size=(3, 3),
kernel_initializer=VarianceScaling(),
use_bias=False,
trainable=False
))
model.add(BatchNormalization(axis=-1))
model.add(Activation('relu'))
model.add(MaxPooling2D(
pool_size=(2, 2),
strides=(2, 2)
))
model.add(Conv2D(
filters=512,
kernel_size=(3, 3),
kernel_initializer=VarianceScaling(),
kernel_regularizer=regularizers.l2(0.001),
use_bias=False,
trainable=False
))
model.add(BatchNormalization(axis=-1))
model.add(Activation('relu'))
model.add(MaxPooling2D(
pool_size=(2, 2),
strides=(2, 2)
))
model.add(Flatten())
model.add(Dense(
units=512,
kernel_initializer=VarianceScaling(),
activation='relu'
))
model.add(Dense(
units=256,
kernel_initializer=VarianceScaling(),
activation='relu'
))
model.add(Dense(
units=128,
kernel_initializer=VarianceScaling(),
activation='relu'
))
model.add(Dense(
units=64,
kernel_initializer=VarianceScaling(),
activation='relu'
))
model.add(Dense(
units=32,
kernel_initializer=VarianceScaling(),
activation='relu'
))
model.add(Dense(
units=16,
kernel_initializer=VarianceScaling(),
activation='relu'
))
model.add(Dense(
units=7,
kernel_initializer=VarianceScaling(),
activation='softmax'
))
model.compile(
loss='categorical_crossentropy',
optimizer="adam",
metrics=['accuracy']
```





Pokusacemo da zakljucamo conv layere I da nastavimo trening nad denseovima
U prvih par iteracija odmah se malo poboljsao val_acc

```
Epoch 1/200
50/50 [==============================] - 6s 122ms/step - loss: 0.8023 - acc: 0.8120 - val_loss: 1.4885 - val_acc: 0.6403
Epoch 2/200
50/50 [==============================] - 5s 99ms/step - loss: 0.8134 - acc: 0.8069 - val_loss: 1.4264 - val_acc: 0.6451
Epoch 3/200
50/50 [==============================] - 5s 97ms/step - loss: 0.7651 - acc: 0.8225 - val_loss: 1.5053 - val_acc: 0.6471
Epoch 4/200
50/50 [==============================] - 5s 95ms/step - loss: 0.7950 - acc: 0.8158 - val_loss: 1.5222 - val_acc: 0.6363
Epoch 5/200
50/50 [==============================] - 5s 104ms/step - loss: 0.7798 - acc: 0.8191 - val_loss: 1.4964 - val_acc: 0.6419
Epoch 6/200
50/50 [==============================] - 5s 97ms/step - loss: 0.7634 - acc: 0.8217 - val_loss: 1.5499 - val_acc: 0.6440
Epoch 7/200
50/50 [==============================] - 5s 97ms/step - loss: 0.7661 - acc: 0.8223 - val_loss: 1.5077 - val_acc: 0.6410
Epoch 8/200
50/50 [==============================] - 5s 97ms/step - loss: 0.7673 - acc: 0.8248 - val_loss: 1.5227 - val_acc: 0.6420
Epoch 9/200
50/50 [==============================] - 5s 100ms/step - loss: 0.7687 - acc: 0.8223 - val_loss: 1.4808 - val_acc: 0.6394
Epoch 10/200
50/50 [==============================] - 5s 94ms/step - loss: 0.7572 - acc: 0.8255 - val_loss: 1.5059 - val_acc: 0.6527
Epoch 11/200
50/50 [==============================] - 5s 98ms/step - loss: 0.7668 - acc: 0.8256 - val_loss: 1.4796 - val_acc: 0.6423
Epoch 12/200
50/50 [==============================] - 5s 100ms/step - loss: 0.7559 - acc: 0.8240 - val_loss: 1.4777 - val_acc: 0.6461
```



Pokusacemo da zakljucamo prvi dense layer

Nema promene

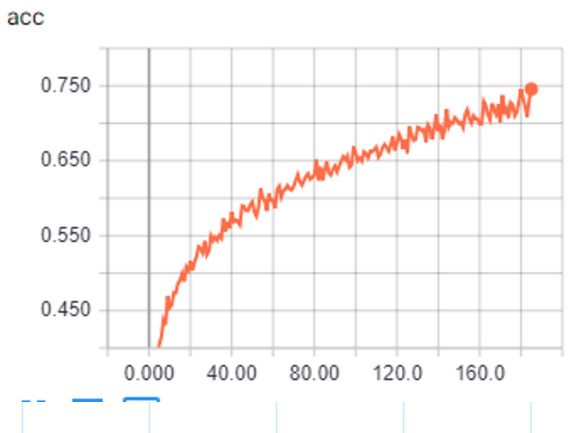
Pokusacemo da odkljucamo konv layere da vidimo sta ce da se desi

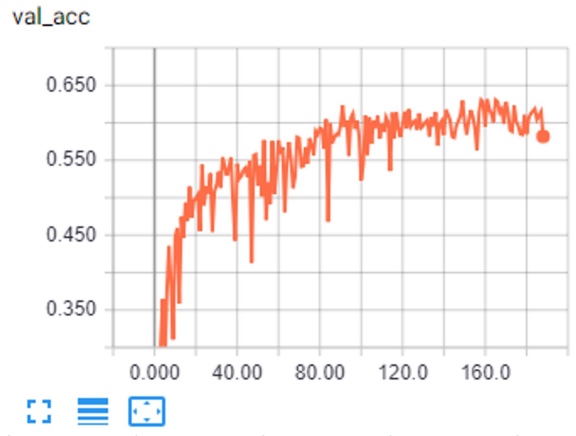
Val_acc pao sa 0.65 na 0.6

```
)
```

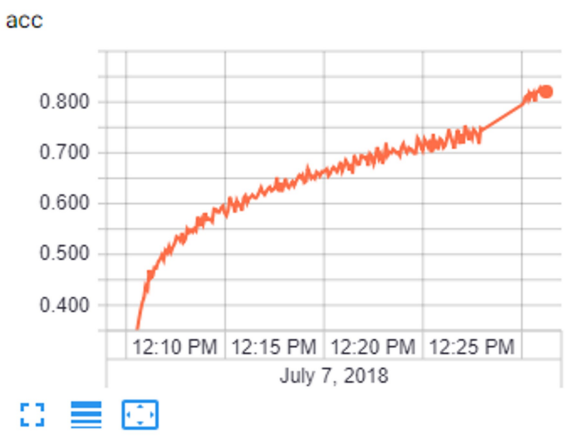
Isti model kao iznad, povecan broj filtera u drugoj konvoluciji

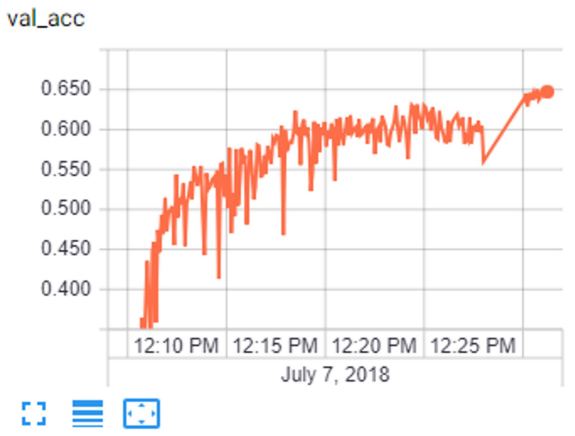
```
model = Sequential()
model.add(Conv2D(
input_shape=(48, 48, 1),
filters=64,
kernel_size=(3, 3),
kernel_initializer=VarianceScaling(),
activation='relu'
))
model.add(MaxPooling2D(
pool_size=(2, 2),
strides=(2, 2)
))
model.add(Conv2D(
filters=256, <------------ 128->256
kernel_size=(3, 3),
kernel_initializer=VarianceScaling(),
use_bias=False
))
model.add(BatchNormalization(axis=-1))
model.add(Activation('relu'))
model.add(MaxPooling2D(
pool_size=(2, 2),
strides=(2, 2)
))
model.add(Conv2D(
filters=512,
kernel_size=(3, 3),
kernel_initializer=VarianceScaling(),
use_bias=False
))
model.add(BatchNormalization(axis=-1))
model.add(Activation('relu'))
model.add(MaxPooling2D(
pool_size=(2, 2),
strides=(2, 2)
))
model.add(Conv2D(
filters=512,
kernel_size=(3, 3),
kernel_initializer=VarianceScaling(),
kernel_regularizer=regularizers.l2(0.001),
use_bias=False
))
model.add(BatchNormalization(axis=-1))
model.add(Activation('relu'))
model.add(MaxPooling2D(
pool_size=(2, 2),
strides=(2, 2)
))
model.add(Flatten())
model.add(Dense(
units=512,
kernel_initializer=VarianceScaling(),
activation='relu'
))
model.add(Dense(
units=256,
kernel_initializer=VarianceScaling(),
activation='relu'
))
model.add(Dense(
units=128,
kernel_initializer=VarianceScaling(),
activation='relu'
))
model.add(Dense(
units=64,
kernel_initializer=VarianceScaling(),
activation='relu'
))
model.add(Dense(
units=32,
kernel_initializer=VarianceScaling(),
activation='relu'
))
model.add(Dense(
units=16,
kernel_initializer=VarianceScaling(),
activation='relu'
))
model.add(Dense(
units=7,
kernel_initializer=VarianceScaling(),
activation='softmax'
))
model.compile(
loss='categorical_crossentropy',
optimizer="adam",
metrics=['accuracy']
)
```



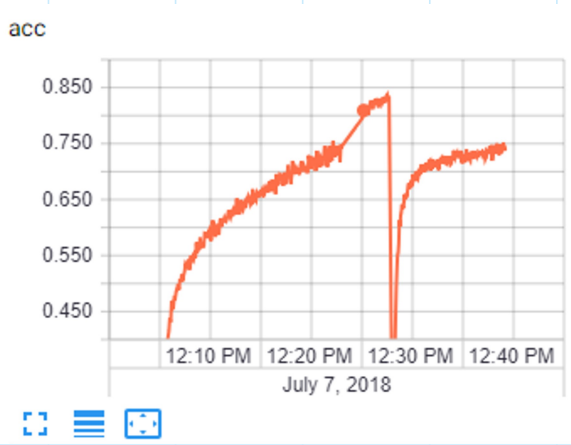


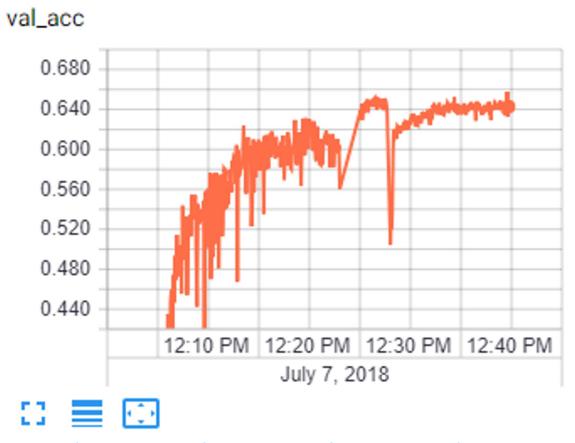
Nakon zakljucavanja conv mreza



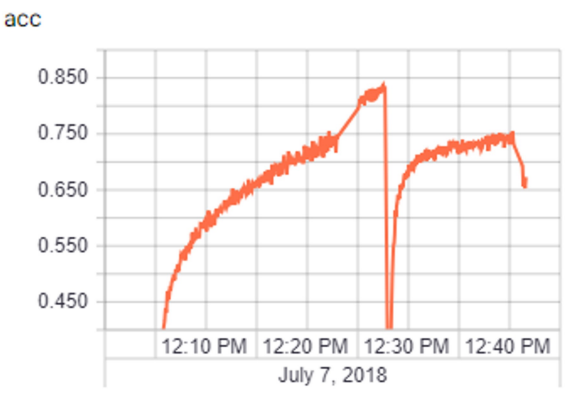


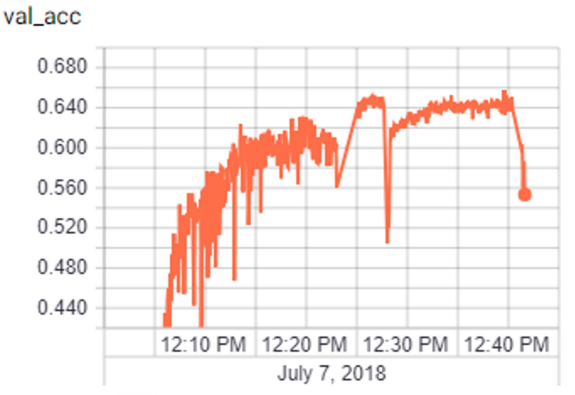
Dodali smo dropout na dense layer nakon treniranja, rezultat ovoga je bio sledeci (validation accuracy je konzistentniji)





Nakon ovoga, pokusali smo da odlkjucamo konvolucione mreze, ali je val_acc poceo da opada (predpostavljamo da je do overfittinga)





Kad smo videli da val_acc opada, prekinuli smo treniranje da bi probali nesto drugo.
Ovog puta stavljamo dropout na konv mreze I odkljucavamo ih.