Duke Classification: with “2D-picture”

Yuhui Cao

20220816 Tue

Roadmap

- Part 1. Convert Threshold to "2D picture"
- Part 2. Shallow Neural Network (SNN)

  - Train on Duke (400)
  - Validate on Duke (100)
  - Test on UAB (301)

- Part 3: Transfer study of **VGG16**

  - Train on Duke (400)
  - Validate on Duke (100)
  - Test on UAB (301)1a

Method 1 - pad with 0

Method 2 – Up-sample smaller img

Part 1. Convert Threshold to “2D picture”

Test Pattern (indexed)

Right Eye

---- index of VF columns

---- index of 10X10 array

	1	2	3	4	5	6	7	8	9	10	
S1	0	1	2	1	2	3	4	7	8	9	27
S2	10		5	6	7	8	9	10		19	21
S3	20	11	12	13	14	15	16	17	18	29	15
S4	19	20	21	22	23	24	25	26	27	28	9
S5	29	30	31	32	33	34	35	36	37	38	3
S6	39	40	41	42	43	44	45	46	47	48	-3
S7	49	50	51	52	53	54	55	56	57	58	-9
S8	70	59	60	61	62	63	64	65	66	79	-15
S9	80		67	68	69	70	71	72		89	-21
S10	90	91	92	73	74	75	76	97	98	99	-27
	-27	-21	-15	-9	-3	3	9	15	21	27	

```
array([[ 0.,  0.,  0., nan, nan, nan, nan,  0.,  0.,  0.],
       [ 0.,  0., nan, 28., 28., 26., 29., nan,  0.,  0.],
       [ 0., nan, 25., 29., 31., 29., 29., 30., nan,  0.],
       [nan, 27., 30., 32., 31., 32., 31., 30., 27., nan],
       [24., 30., 31., 32., 33., 33., 34., 27., 27., nan],
       [22., 27., 30., 32., 34., 32., 32., 16., 27., nan],
       [nan, 21., 23., 24., 30., 33., 31., 27., 25., nan],
       [ 0., nan, 16., 23., 29., 31., 27., 26., nan,  0.],
       [ 0.,  0., nan, 11., 24., 27., 29., nan,  0.,  0.],
       [ 0.,  0.,  0., nan, nan, nan, nan,  0.,  0.,  0.]])
```

Yuhui:

- **Fill the rest of (100-76 =) 24 locations with 0?**
- **Fill NaN of 24-2 with 0?**

Part 1. Convert Threshold to “2D picture”

An example of 24-2

```
array([[ 0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.],
       [ 0.,  0.,  0., 28., 28., 26., 29.,  0.,  0.,  0.],
       [ 0.,  0., 25., 29., 31., 29., 29., 30.,  0.,  0.],
       [ 0., 27., 30., 32., 31., 32., 31., 30., 27.,  0.],
       [24., 30., 31., 32., 33., 33., 34., 27., 27.,  0.],
       [22., 27., 30., 32., 34., 32., 32., 16., 27.,  0.],
       [ 0., 21., 23., 24., 30., 33., 31., 27., 25.,  0.],
       [ 0.,  0., 16., 23., 29., 31., 27., 26.,  0.,  0.],
       [ 0.,  0.,  0., 11., 24., 27., 29.,  0.,  0.,  0.],
       [ 0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.]])
```

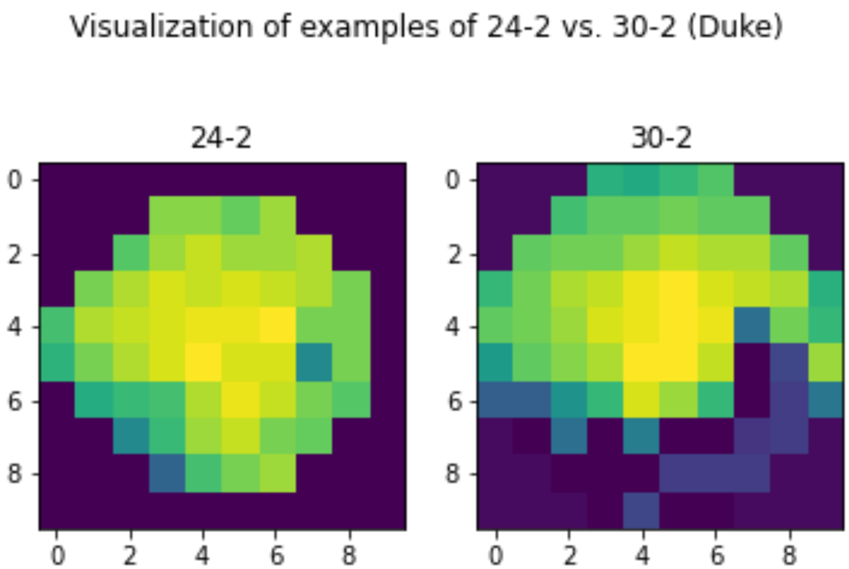
An example of 30-2

```
array([[ 0.,  0.,  0., 20., 19., 21., 23.,  0.,  0.,  0.],
       [ 0.,  0., 22., 24., 24., 25., 24., 24.,  0.,  0.],
       [ 0., 24., 25., 25., 27., 29., 28., 28., 24.,  0.],
       [21., 25., 28., 29., 31., 32., 30., 29., 28., 20.],
       [24., 25., 27., 30., 31., 32., 31., 11., 25., 21.],
       [17., 24., 26., 28., 32., 32., 29., -1.,  6., 27.],
       [ 9.,  9., 16., 21., 30., 27., 21., -1.,  5., 12.],
       [ 0., -1., 11., -1., 13., -1., -1.,  4.,  5.,  0.],
       [ 0.,  0., -1., -1., -1.,  5.,  5.,  5.,  0.,  0.],
       [ 0.,  0.,  0., -1.,  6., -1., -1.,  0.,  0.,  0.]])
```

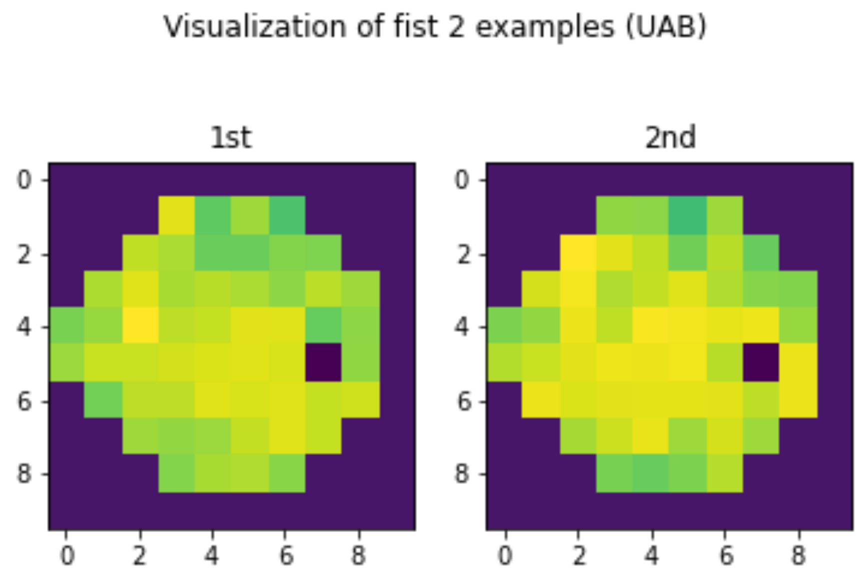
500 cases:

- The number of 30-2 pattern is: 14
- The number of 24-2 pattern is: 486

Part 1. Convert Threshold to “2D picture”



500 Duke



301 UAB

Part 2. Shallow Neural Network (SNN)

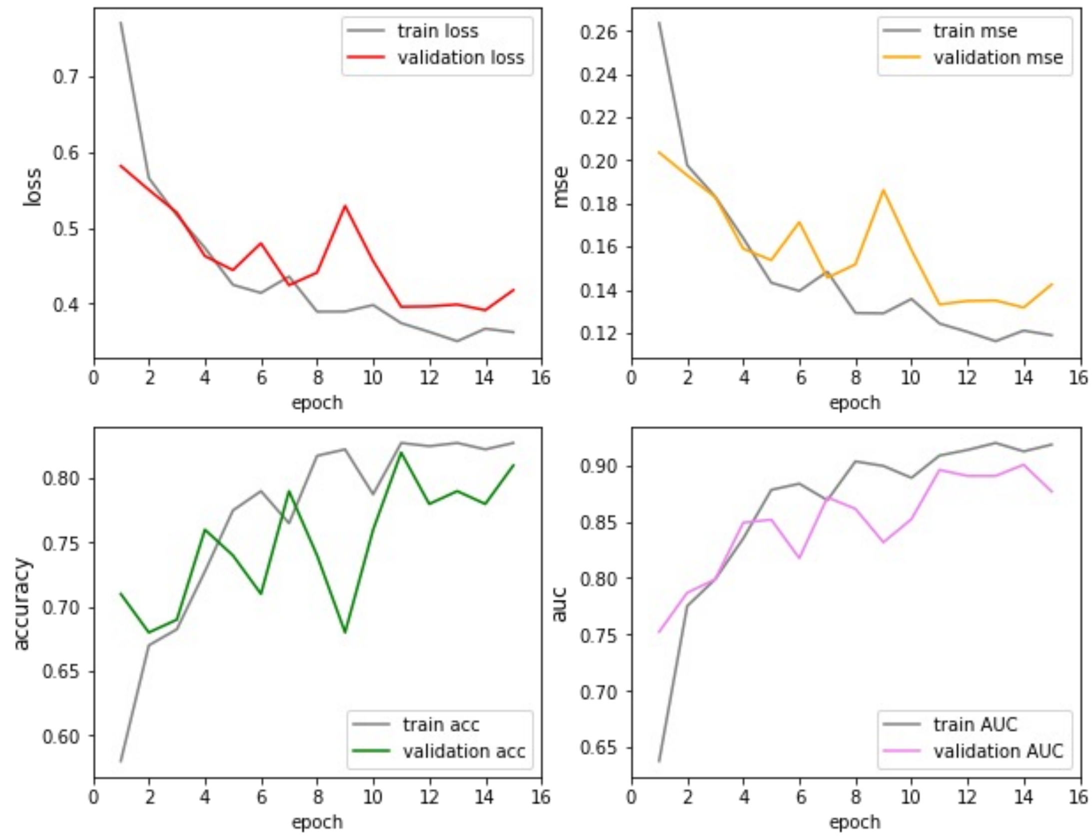
Classification on normal vs. abnormal

SNN 1

```
# create model
model = Sequential()
model.add(Conv2D(16, (2, 2), strides=(1, 1), activation='relu',
                 input_shape=(10, 10, 1)))
model.add(MaxPooling2D(pool_size=(2, 2), strides=(2, 2)))

model.add(Flatten())
model.add(Dense(20, activation='relu'))
model.add(Dense(num_classes, activation='softmax'))
```

epochs = 15
batch_size=30



Part 2. Shallow Neural Network (SNN)

Classification on normal vs. abnormal

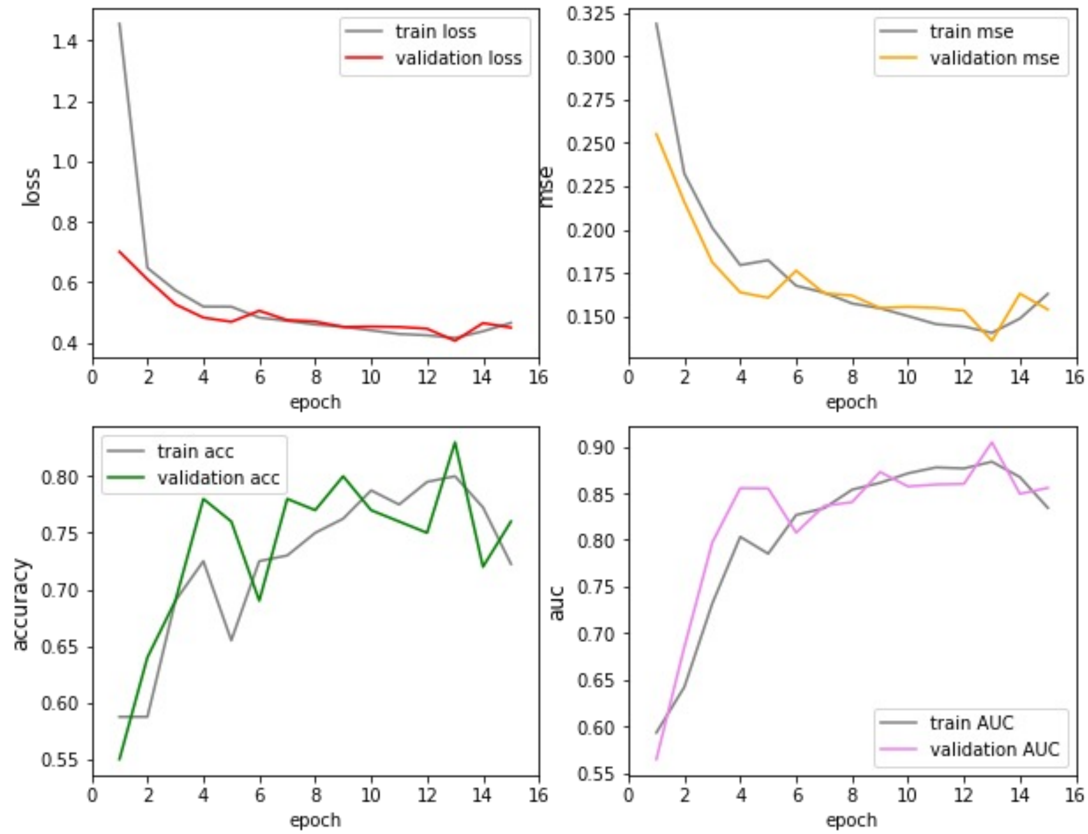
SNN 2

```
# create model
model = Sequential()
model.add(Conv2D(16, (2, 2), strides=(1, 1), activation='relu',
                 input_shape=(10, 10, 1)))
model.add(MaxPooling2D(pool_size=(2, 2), strides=(2, 2)))

model.add(Conv2D(8, (2, 2), activation='relu'))
#model.add(MaxPooling2D(pool_size=(2, 2), strides=(2, 2)))

model.add(Flatten())
model.add(Dense(20, activation='relu'))
model.add(Dense(num_classes, activation='softmax'))
```

epochs = 15
batch_size=30



Part 2. Shallow Neural Network (SNN)

Classification on normal vs. abnormal

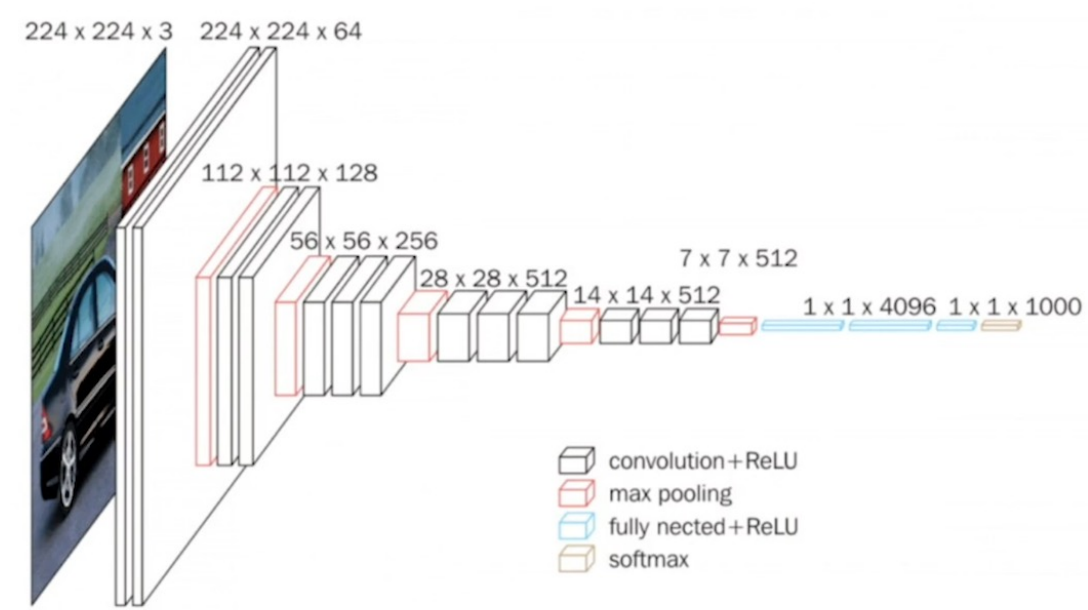
	validation (M1)	test (M1)	validation (M2)	test (M2)
loss	0.417800	0.582213	0.450455	0.563872
mse	0.142439	0.212142	0.153874	0.201238
accuracy	0.810000	0.637874	0.760000	0.657807
auc	0.876650	0.756371	0.855800	0.742376

Validation on Duke (100)
Test on UAB (301)

```
model1.save("SNN_1.h5")
model2.save("SNN_2.h5")
```

Part 3: Transfer study of VGG16

CNNs



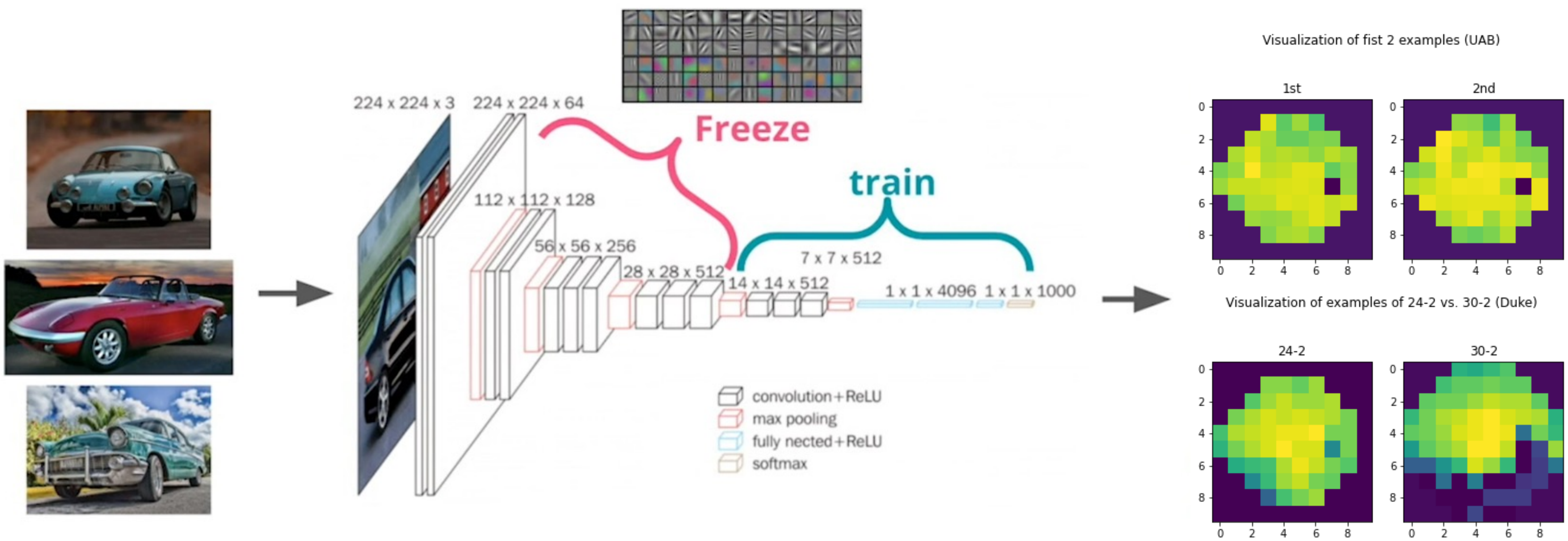
Yuhui strategies:

1. **Should I start from 14X14X512 layer? (can very easily pad 10X10 to 14X14)**
2. **Resize 10X0 to 224X224 so that can use earlier layer? (preferred)**

Part 3: Transfer study of VGG16

Fine-tuning

Repurposing pre-trained architectures



Part 3: Transfer study of VGG16

Dimension construction: resize to 224 X 224

1. 10 X 10
2. 224 X 224
3. 224 X 224 X 3
   - repeat **thr** 3 times
   - future: use **thr**, **TD**, **PD**
4. 500 X 224 X 224

Method 1 - pad with 0

Method 2 – Up-sample smaller img

Method 3 – Interpolating smaller img

Result 0. Pad with NaN is not better ~

```
Epoch 1/5
16/16 [==============================] - 174s 11s/step - loss: nan - binary_accuracy: 0.630
0 - val_loss: nan - val_binary_accuracy: 0.8472
Epoch 2/5
16/16 [==============================] - 188s 12s/step - loss: nan - binary_accuracy: 0.630
0 - val_loss: nan - val_binary_accuracy: 0.8472
Epoch 3/5
16/16 [==============================] - 193s 12s/step - loss: nan - binary_accuracy: 0.630
0 - val_loss: nan - val_binary_accuracy: 0.8472
Epoch 4/5
16/16 [==============================] - 177s 11s/step - loss: nan - binary_accuracy: 0.630
0 - val_loss: nan - val_binary_accuracy: 0.8472
Epoch 5/5
16/16 [==============================] - 184s 12s/step - loss: nan - binary_accuracy: 0.630
0 - val_loss: nan - val_binary_accuracy: 0.8472
```

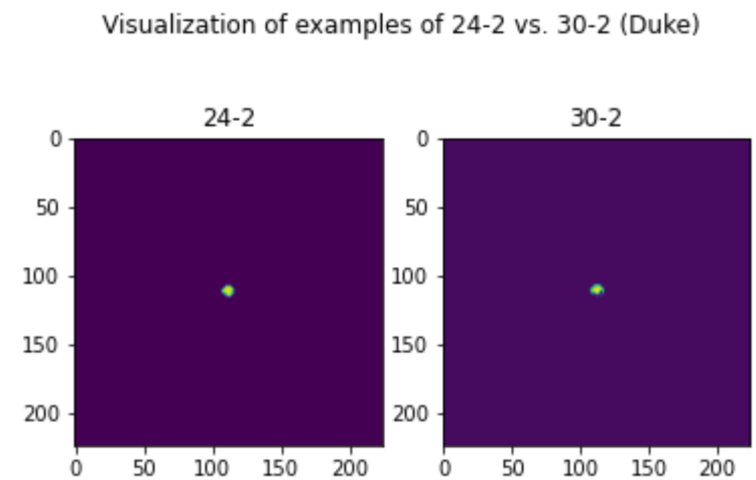
Result 1. Method 1 - pad with 0

Model 1: a simple sequential model using only the VGG16 architecture

Model 2: add a few more dense layers

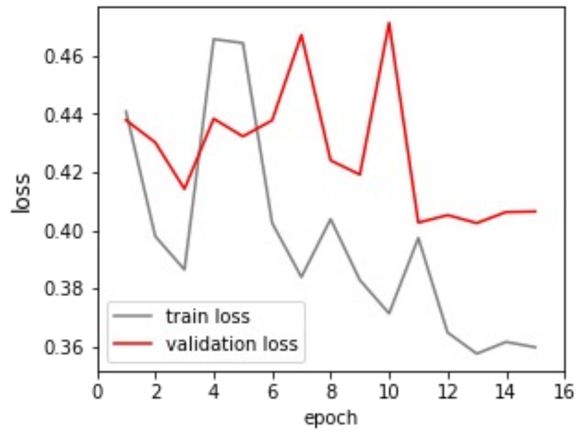
Model 3: add dropout and another fully connected layer

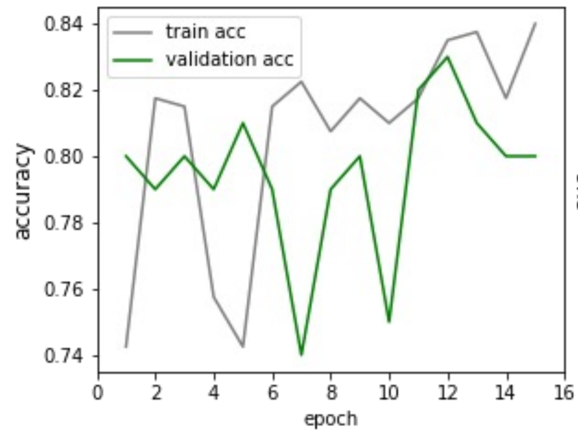
Image Visualization

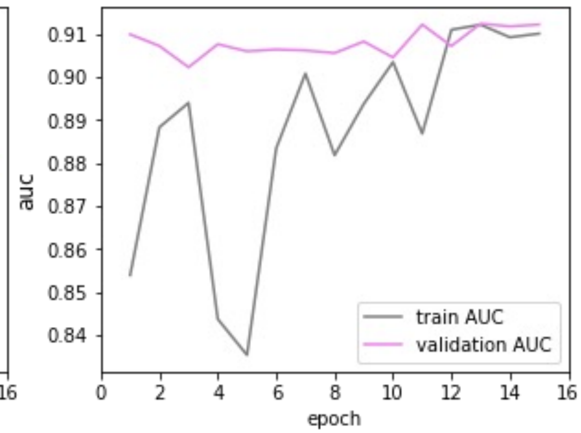


Part 3: Transfer study of VGG16 – model 1

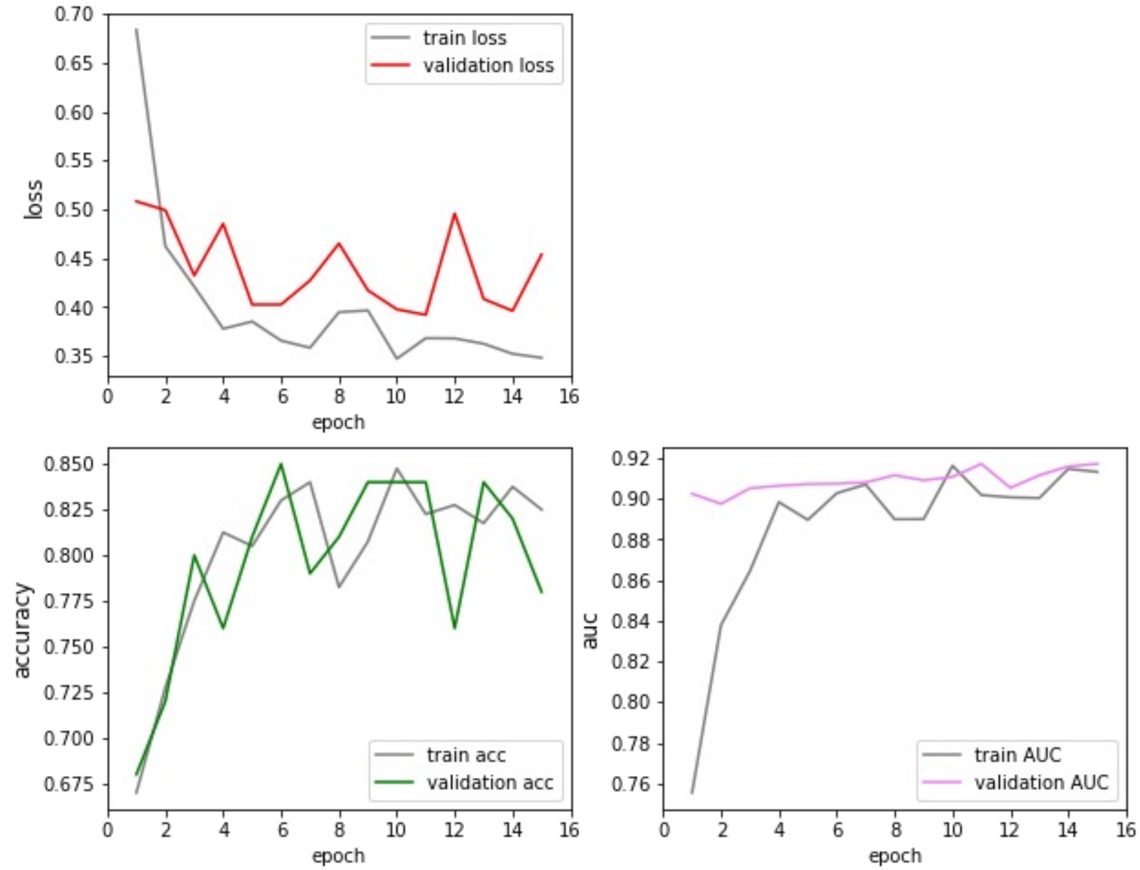
epochs = 15
batch_size = 32 (default)





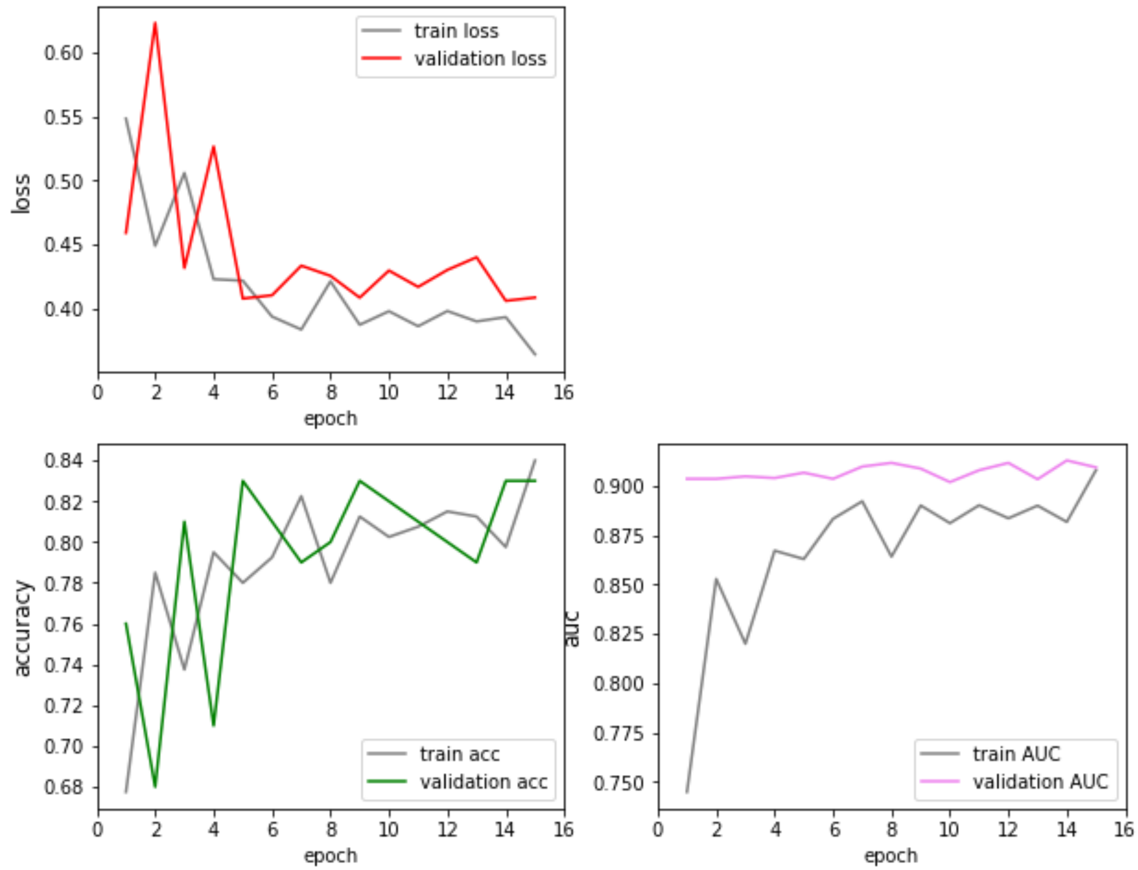


Part 3: Transfer study of VGG16 – model 2



epochs = 15
batch_size = 32 (default)

Part 3: Transfer study of VGG16 – model 3



epochs = 15
batch_size = 32 (default)

Test: Method 1 - pad with 0 (on UAB)

	validation (M1)	test (M1)	validation (M2)	test (M2)	validation (M3)	test (M3)
loss	0.406474	0.590327	0.453722	0.301917	0.408702	0.412213
accuracy	0.800000	0.651163	0.780000	0.857143	0.830000	0.777409
auc	0.912154	0.902899	0.917114	0.911978	0.909260	0.900384

Validation on Duke (100)
Test on UAB (301)

```
model1.save("VGG1.h5")
model2.save("VGG2.h5")
model3.save("VGG3.h5")
```

Result 2. Method 2 – Up-sample smaller img

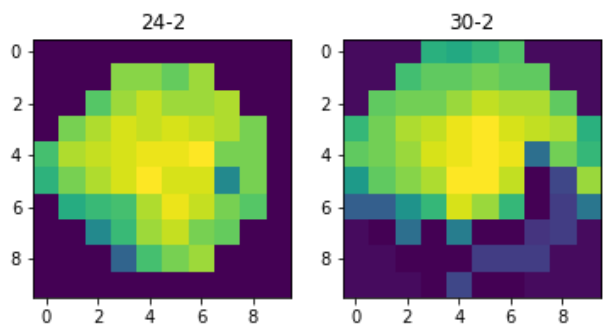
Model 1: a simple sequential model using only the VGG16 architecture

Model 2: add a few more dense layers

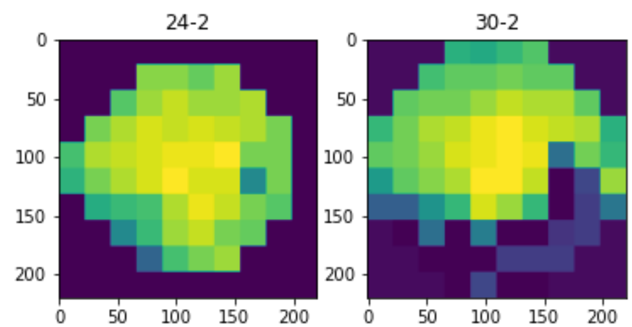
Model 3: add dropout and another fully connected layer

Image Visualization

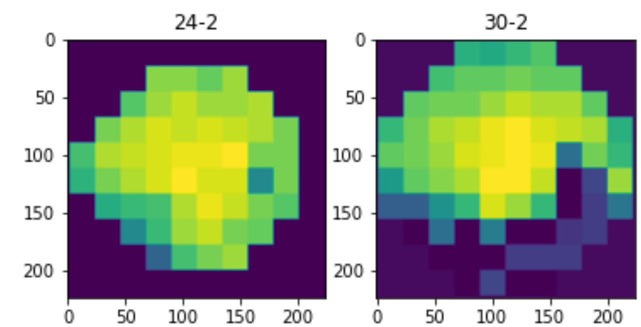
1. Visualization of examples of 24-2 vs. 30-2 (Duke) [10X10]



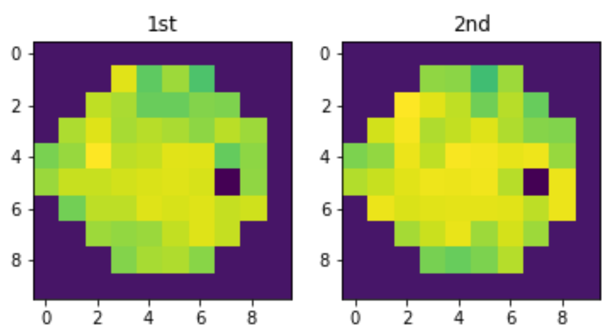
2. Visualization of examples of 24-2 vs. 30-2 (Duke) [220X220]



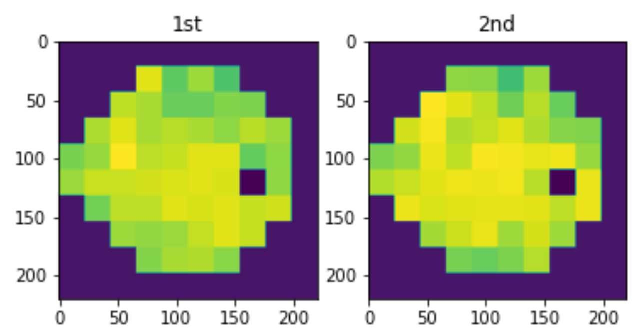
3. Visualization of examples of 24-2 vs. 30-2 (Duke) [224X224]



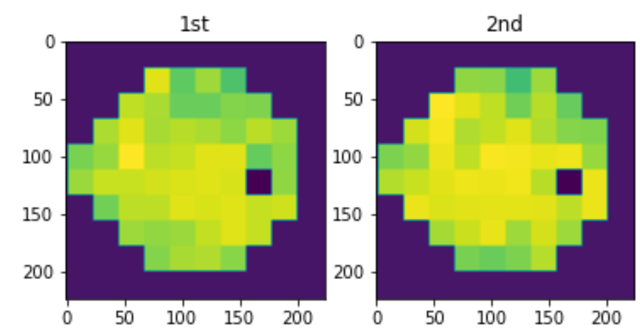
1. Visualization of fist 2 examples (UAB) [10X10]



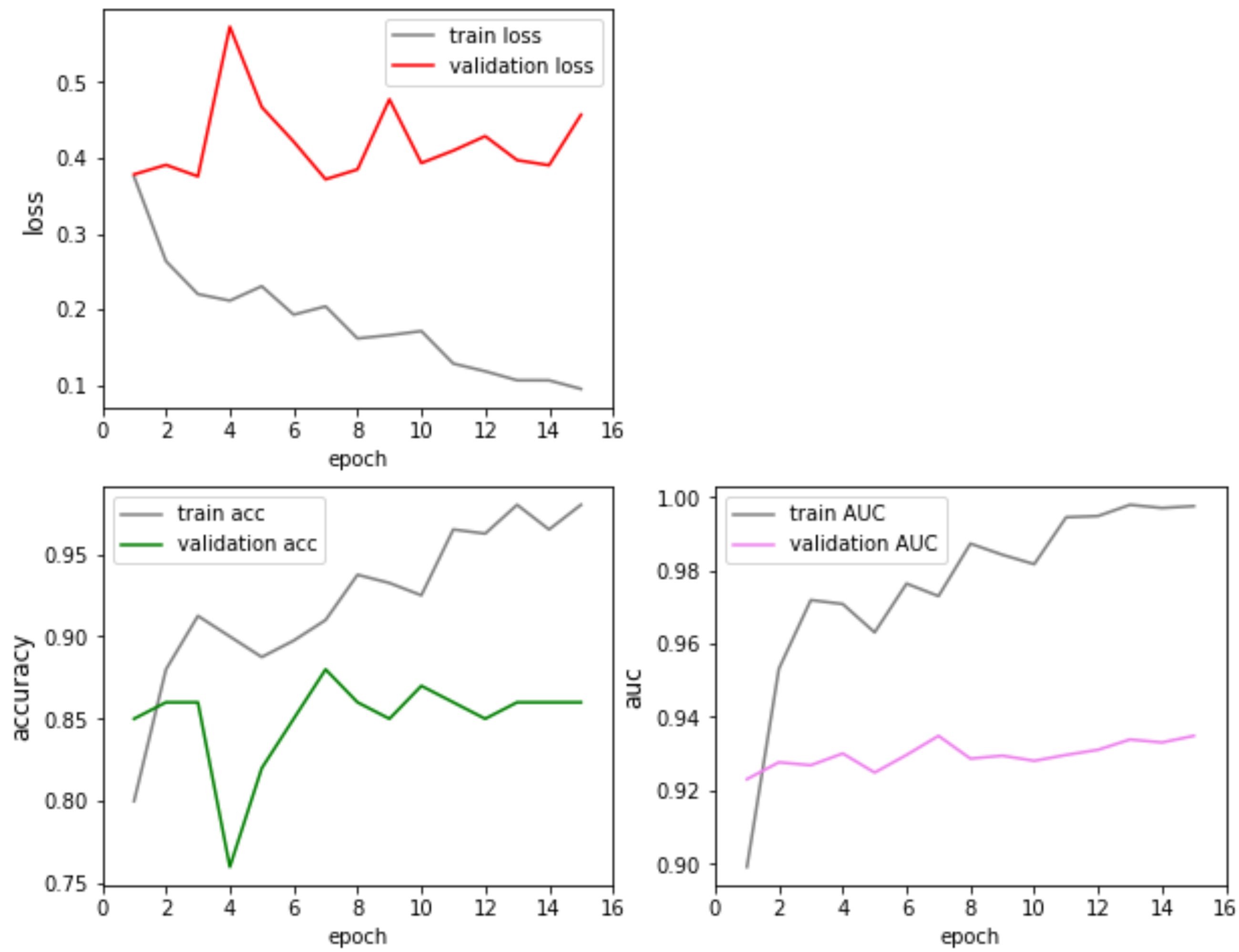
2. Visualization of fist 2 examples (UAB) [220X220]



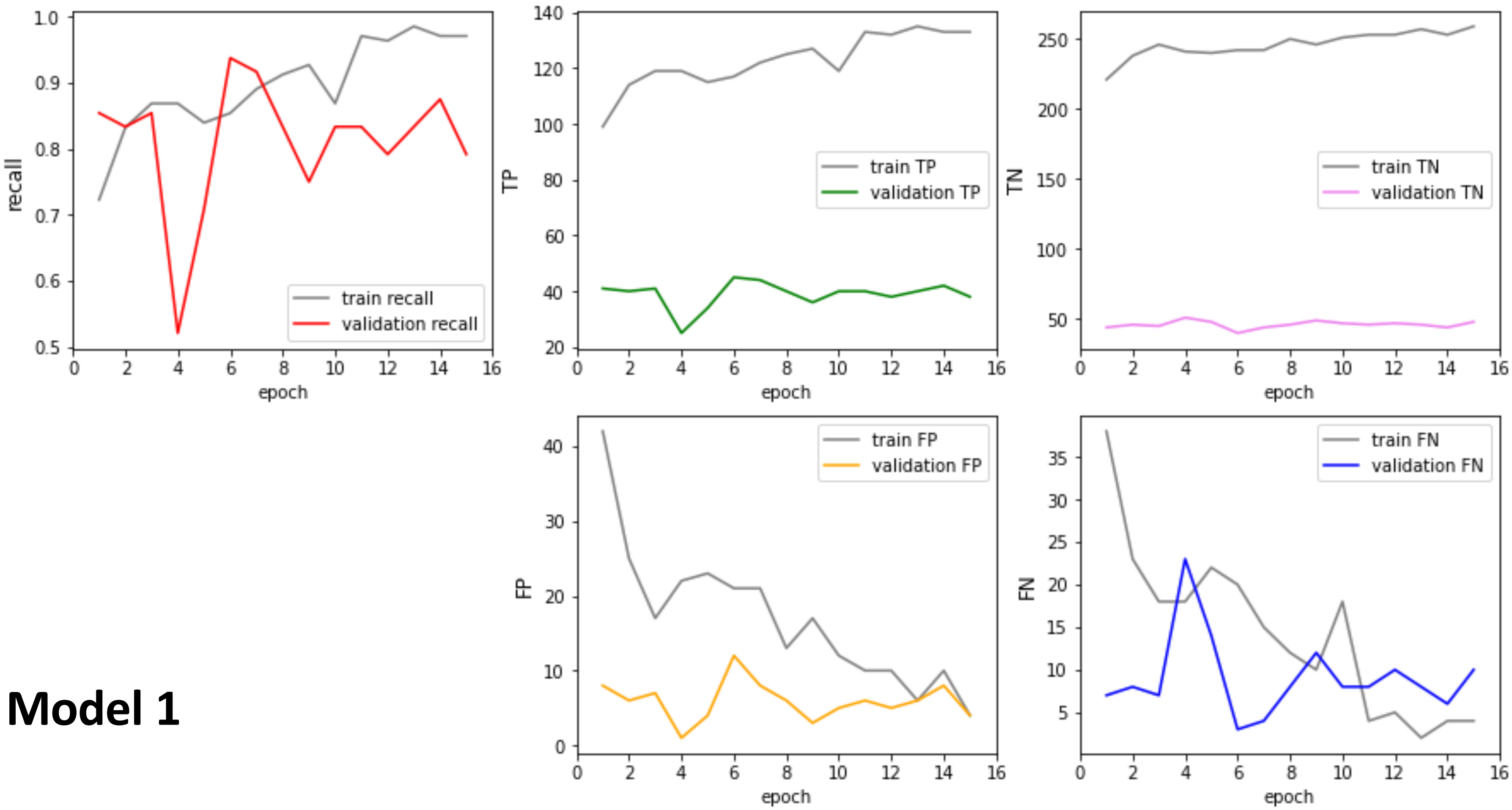
3. Visualization of fist 2 examples (UAB) [224X224]



Model 1

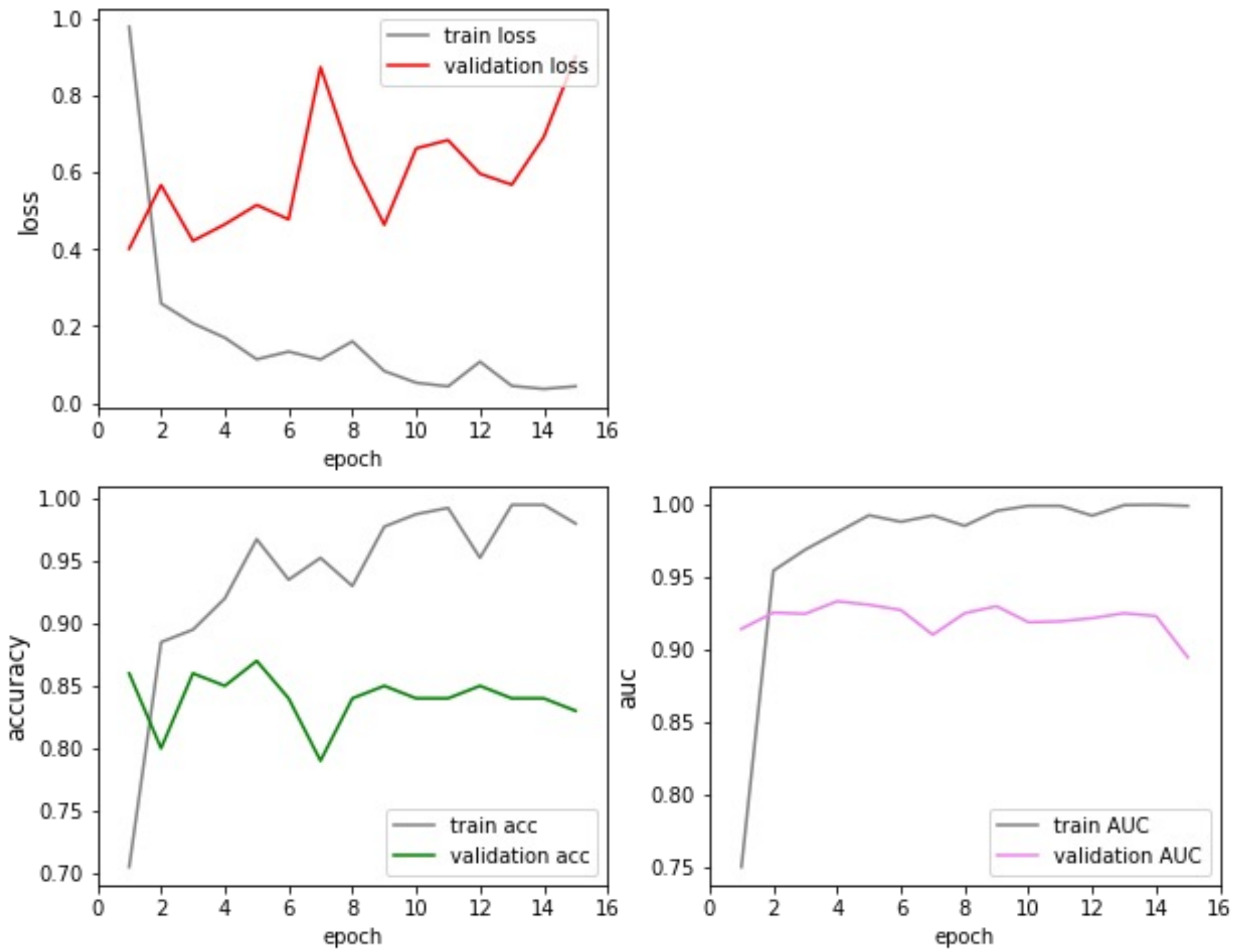


train loss
validation loss
loss
epoch
train acc
validation acc
accuracy
epoch
train AUC
validation AUC
auc
epoch

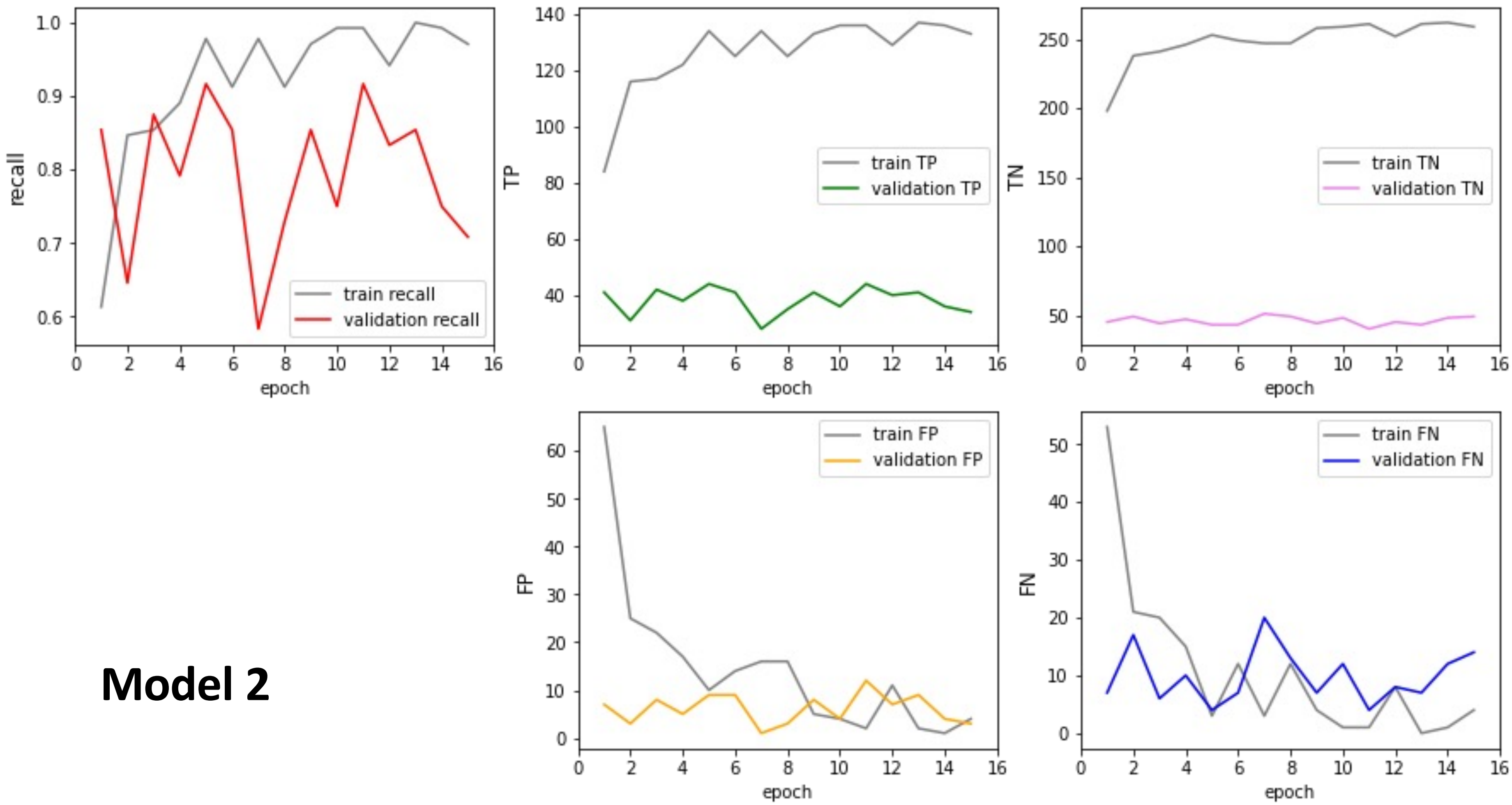

recall
train recall
validation recall
epoch
TP
train TP
validation TP
TN
train TN
validation TN
FP
train FP
validation FP
FN
train FN
validation FN

Model 1

Model 2

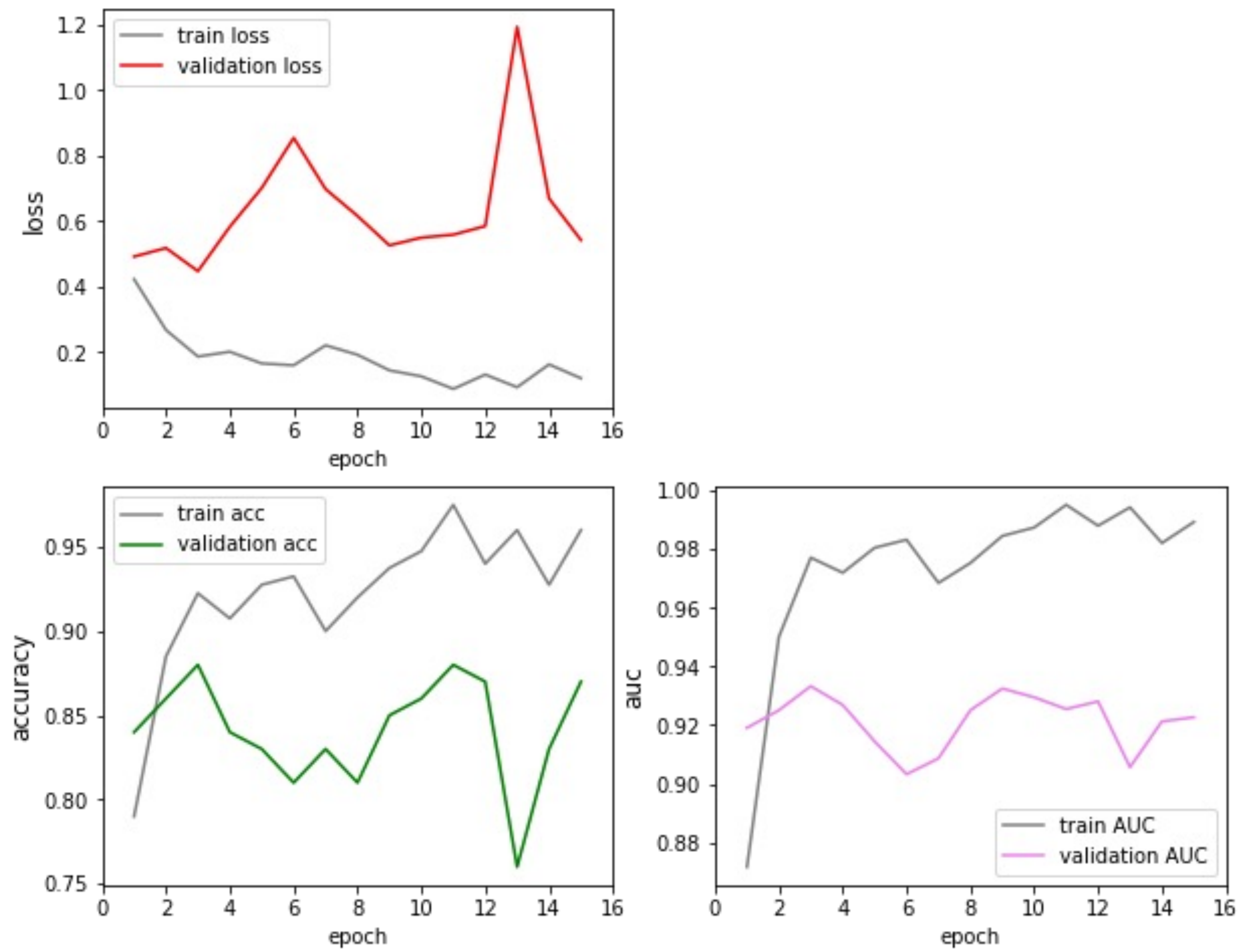


train loss
validation loss
loss
epoch
train acc
validation acc
accuracy
epoch
train AUC
validation AUC
auc
epoch

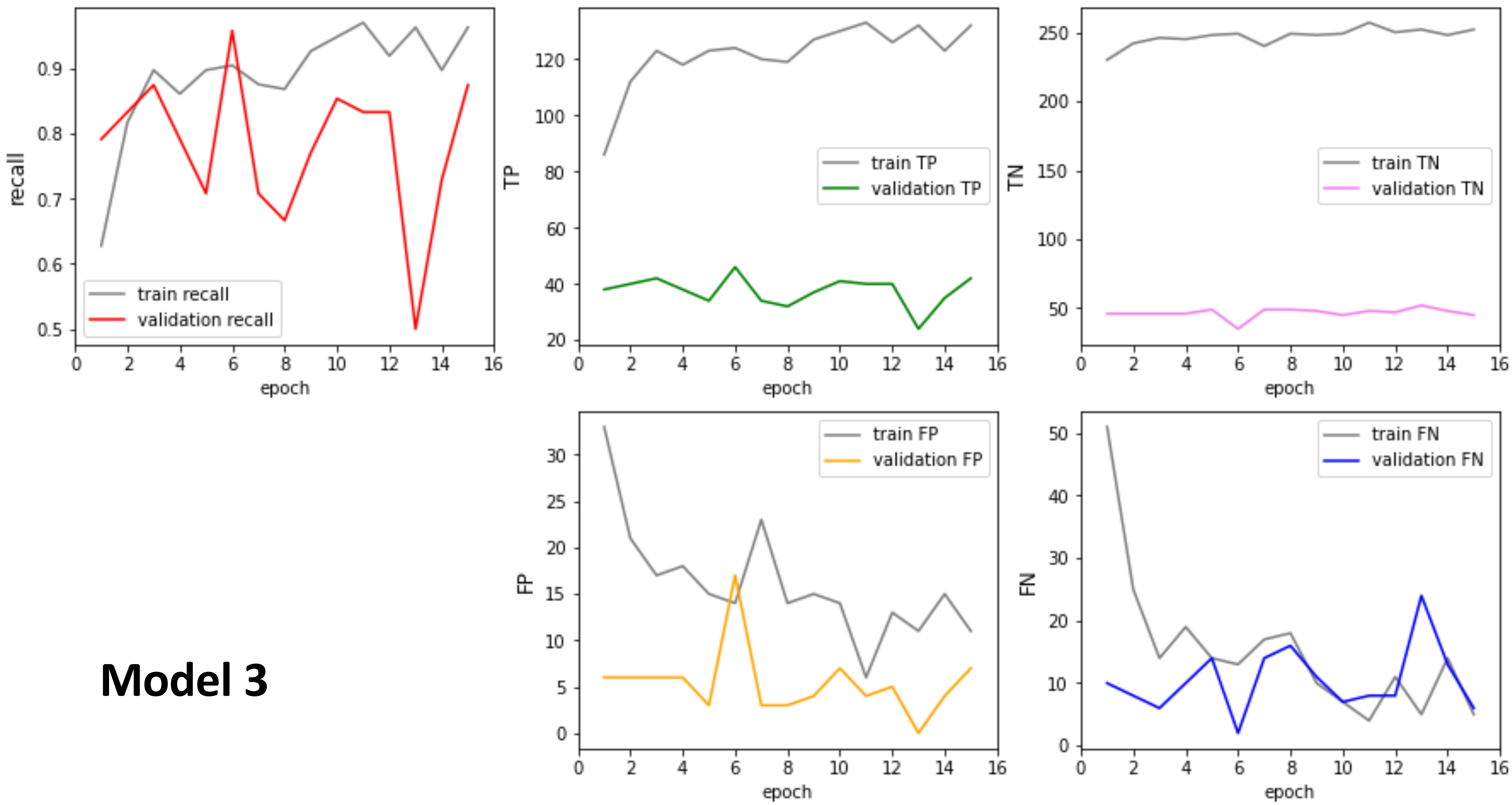

recall
epoch
train recall
validation recall
TP
epoch
train TP
validation TP
TN
epoch
train TN
validation TN
FP
epoch
train FP
validation FP
FN
epoch
train FN
validation FN

Model 2

Model 3



train loss
validation loss
loss
epoch
train acc
validation acc
accuracy
epoch
train AUC
validation AUC
auc
epoch



train recall
validation recall
recall
epoch
train TP
validation TP
TP
epoch
train TN
validation TN
TN
epoch
train FP
validation FP
FP
epoch
train FN
validation FN
FN
epoch

Model 3

Model 1

Train (Duke 400)

	TP	TN	FP	FN	sens	spec
0	99.0	221.0	42.0	38.0	0.722628	0.840304
1	114.0	238.0	25.0	23.0	0.832117	0.904943
2	119.0	246.0	17.0	18.0	0.868613	0.935361
3	119.0	241.0	22.0	18.0	0.868613	0.916350
4	115.0	240.0	23.0	22.0	0.839416	0.912548
5	117.0	242.0	21.0	20.0	0.854015	0.920152
6	122.0	242.0	21.0	15.0	0.890511	0.920152
7	125.0	250.0	13.0	12.0	0.912409	0.950570
8	127.0	246.0	17.0	10.0	0.927007	0.935361
9	119.0	251.0	12.0	18.0	0.868613	0.954373
10	133.0	253.0	10.0	4.0	0.970803	0.961977
11	132.0	253.0	10.0	5.0	0.963504	0.961977
12	135.0	257.0	6.0	2.0	0.985401	0.977186
13	133.0	253.0	10.0	4.0	0.970803	0.961977
14	133.0	259.0	4.0	4.0	0.970803	0.984791

Validation (Duke 100)

	TP_val	TN_val	FP_val	FN_val	sens_val	spec_val
0	41.0	44.0	8.0	7.0	0.854167	0.846154
1	40.0	46.0	6.0	8.0	0.833333	0.884615
2	41.0	45.0	7.0	7.0	0.854167	0.865385
3	25.0	51.0	1.0	23.0	0.520833	0.980769
4	34.0	48.0	4.0	14.0	0.708333	0.923077
5	45.0	40.0	12.0	3.0	0.937500	0.769231
6	44.0	44.0	8.0	4.0	0.916667	0.846154
7	40.0	46.0	6.0	8.0	0.833333	0.884615
8	36.0	49.0	3.0	12.0	0.750000	0.942308
9	40.0	47.0	5.0	8.0	0.833333	0.903846
10	40.0	46.0	6.0	8.0	0.833333	0.884615
11	38.0	47.0	5.0	10.0	0.791667	0.903846
12	40.0	46.0	6.0	8.0	0.833333	0.884615
13	42.0	44.0	8.0	6.0	0.875000	0.846154
14	38.0	48.0	4.0	10.0	0.791667	0.923077

sensitivity, **recall**, **hit rate**, or **true positive rate** (TPR)

$$\mathrm{TPR} = \frac{\mathrm{TP}}{\mathrm{P}} = \frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FN}} = 1 - \mathrm{FNR}$$

specificity, **selectivity** or **true negative rate** (TNR)

$$\mathrm{TNR} = \frac{\mathrm{TN}}{\mathrm{N}} = \frac{\mathrm{TN}}{\mathrm{TN}+\mathrm{FP}} = 1 - \mathrm{FPR}$$

Validation:

- Specificity was OK.
- Sensitivity was bad

Solution:

- increase power
- Increase sample size

Test: Method 2 – Up-sample (on UAB)

	validation (M1)	test (M1)	validation (M2)	test (M2)	validation (M3)	test (M3)
loss	0.296763	0.378864	0.199720	0.395243	0.244206	0.774924
accuracy	0.940000	0.787375	0.970000	0.797342	0.930000	0.730897
auc	0.980035	0.917008	0.979167	0.902856	0.978733	0.909548
recall	0.916667	0.913043	0.944444	0.869565	0.972222	0.934783
TP	33.000000	42.000000	34.000000	40.000000	35.000000	43.000000
TN	61.000000	195.000000	63.000000	200.000000	58.000000	177.000000
FP	3.000000	60.000000	1.000000	55.000000	6.000000	78.000000
FN	3.000000	4.000000	2.000000	6.000000	1.000000	3.000000

Validation on Duke (100)
Test on UAB (301)

```
model1.save("VGG1_upsampling.h5")
model2.save("VGG2_upsampling.h5")
model3.save("VGG3_upsampling.h5")
```

Result 3. Method 3 – Interpolate an image

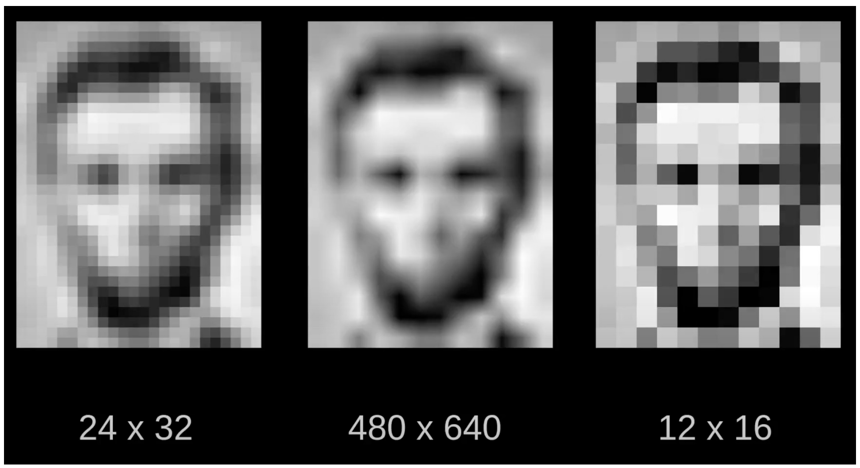
Model 1: a simple sequential model using only the VGG16 architecture

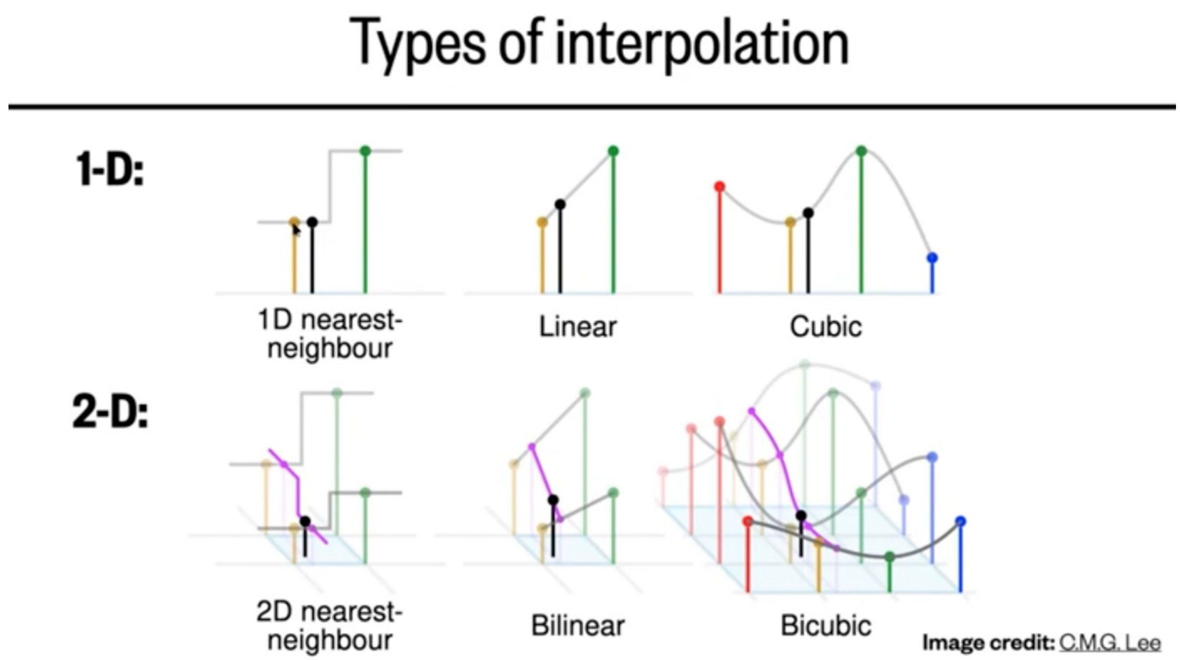
Model 2: add a few more dense layers

Model 3: add dropout and another fully connected layer

Image Processing

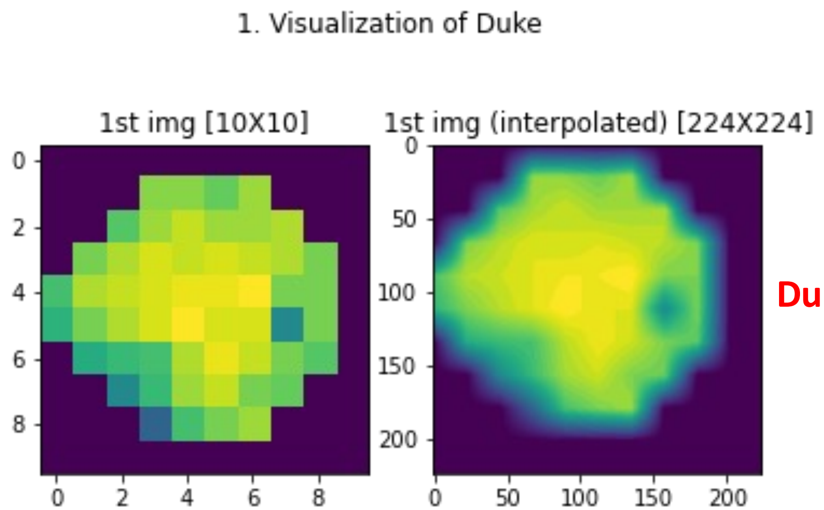
Bilinear Interpolation & Resampling



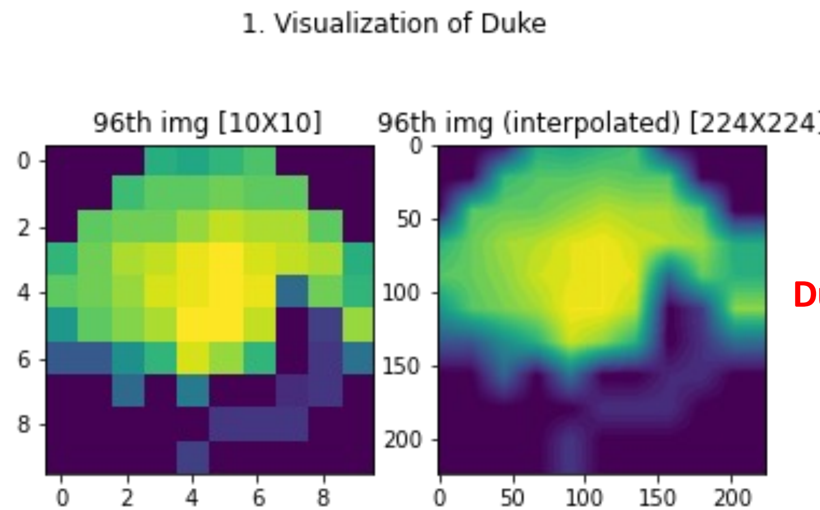


Visualization:

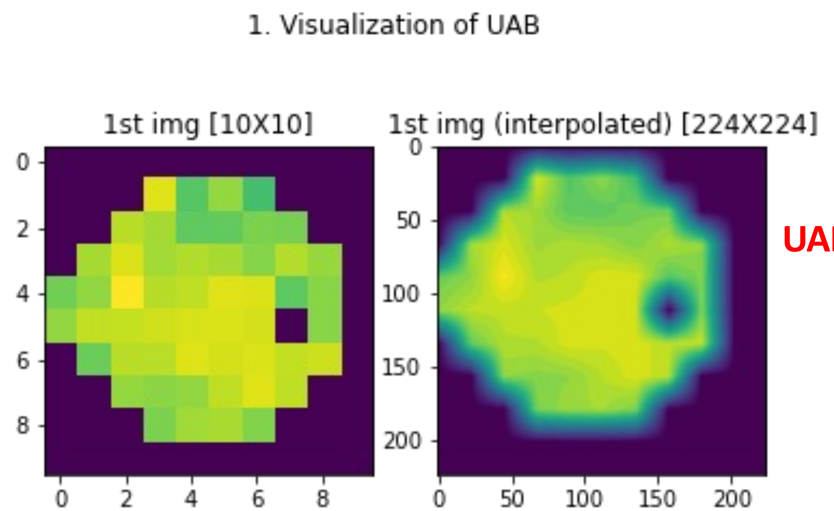
- Original vs. Interpolated



Duke 24-2



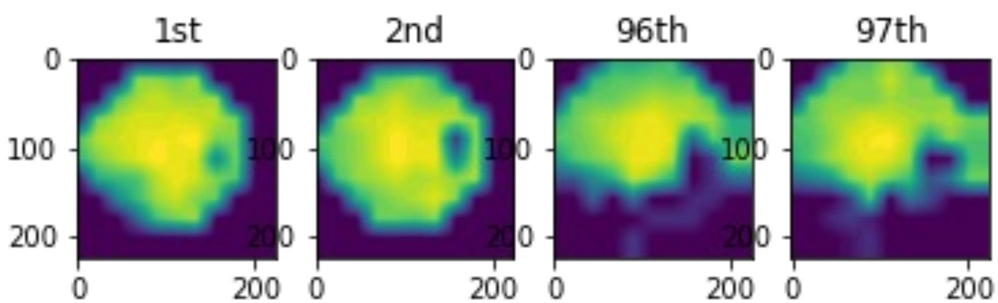
Duke 30-2



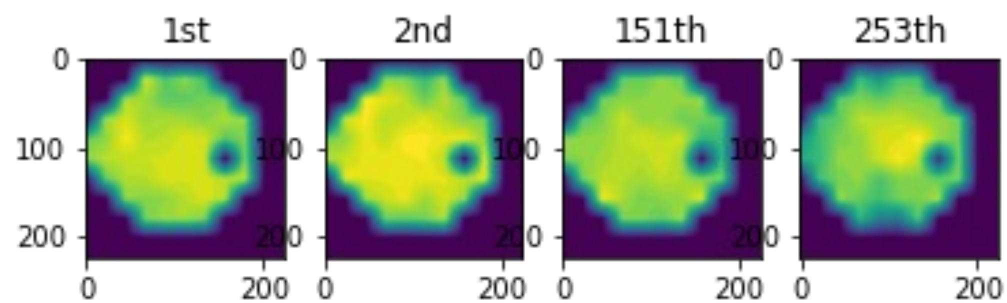
UAB 24-2

Visualization: Interpolated

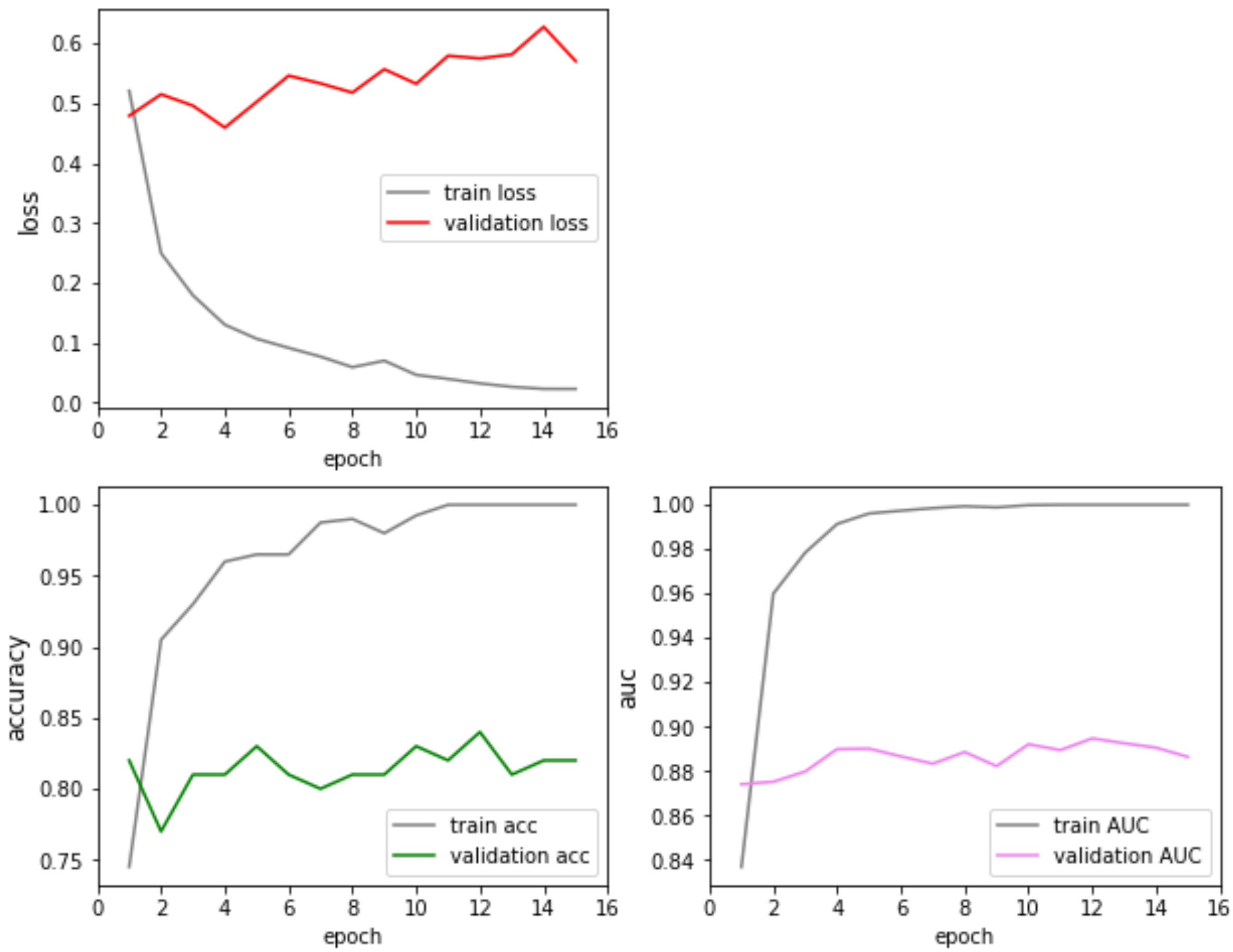
1. Visualization of DUKE [224X24]

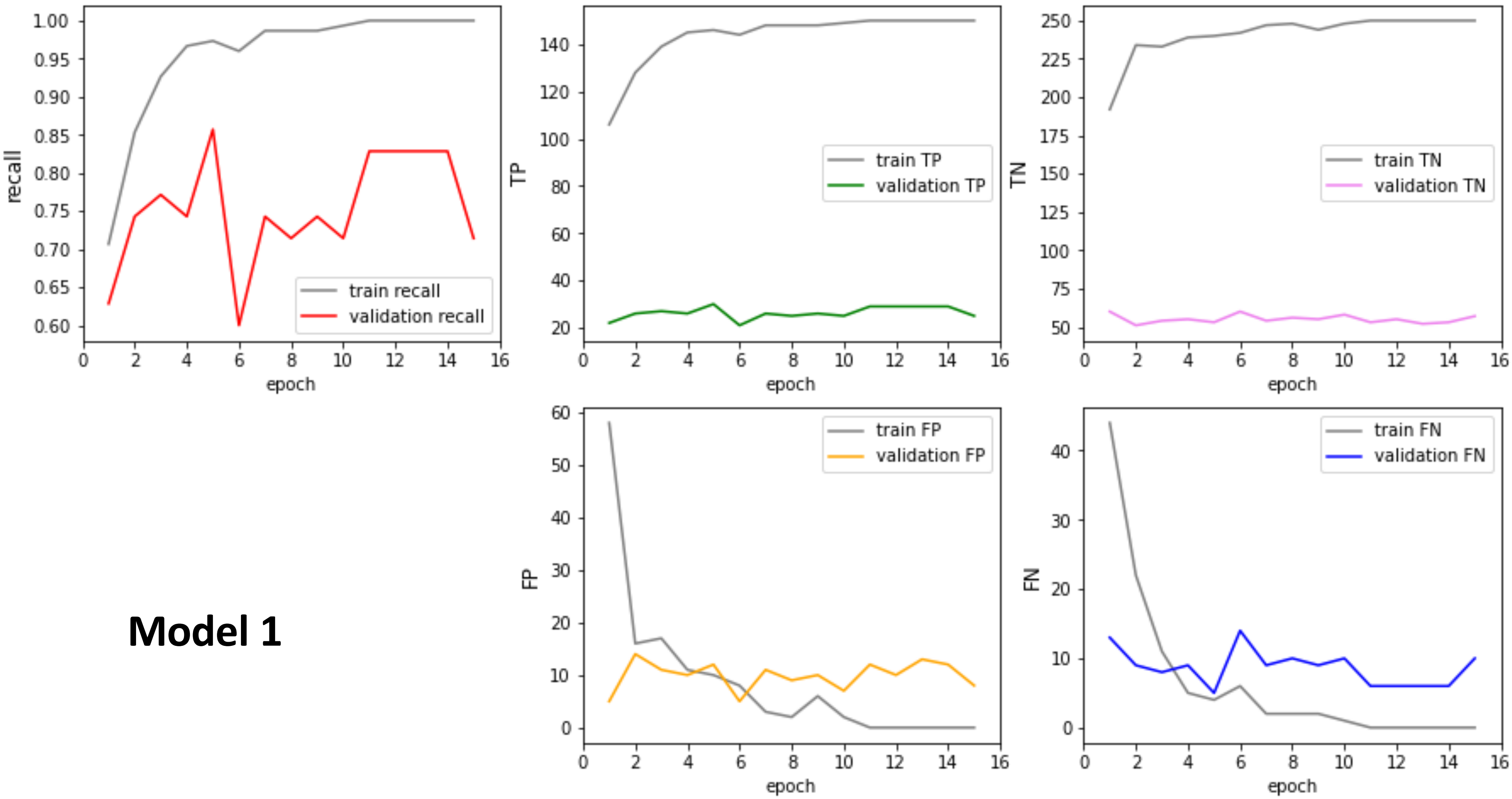


1. Visualization of UAB [224X24]



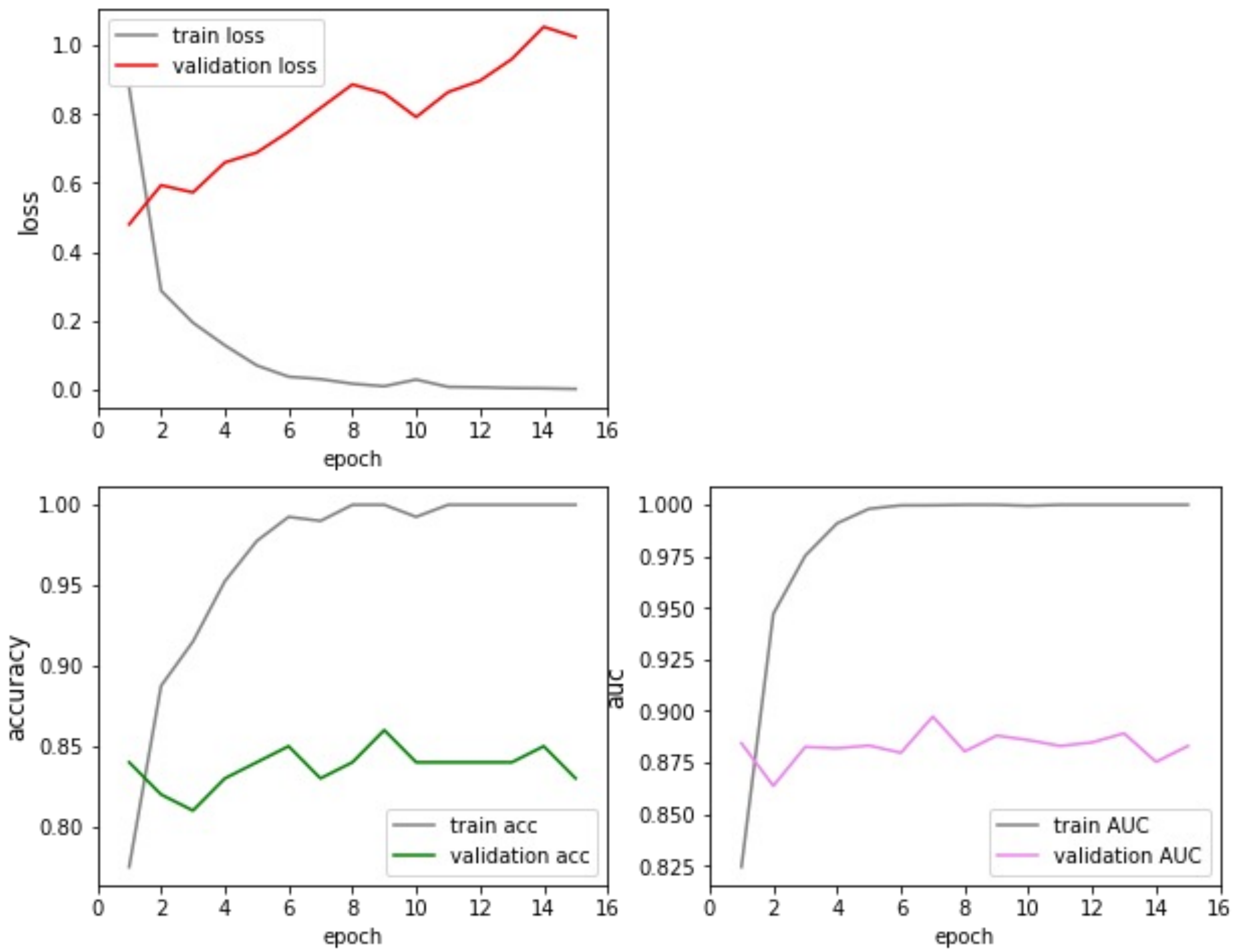
Model 1


loss
epoch
train loss
validation loss
accuracy
epoch
train acc
validation acc
auc
epoch
train AUC
validation AUC

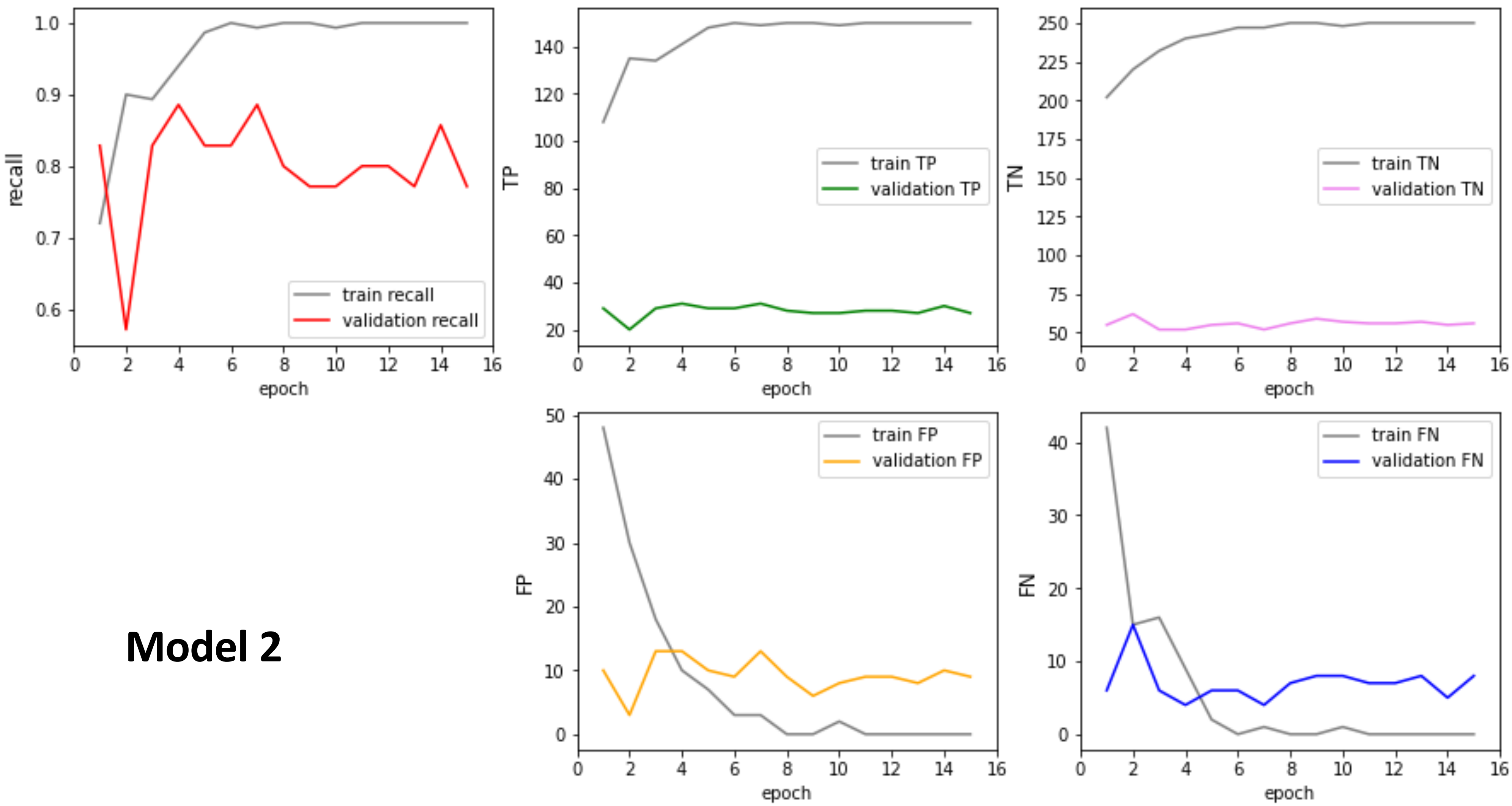

recall
train recall
validation recall
TP
train TP
validation TP
TN
train TN
validation TN
FP
train FP
validation FP
FN
train FN
validation FN
epoch

Model 1

Model 2

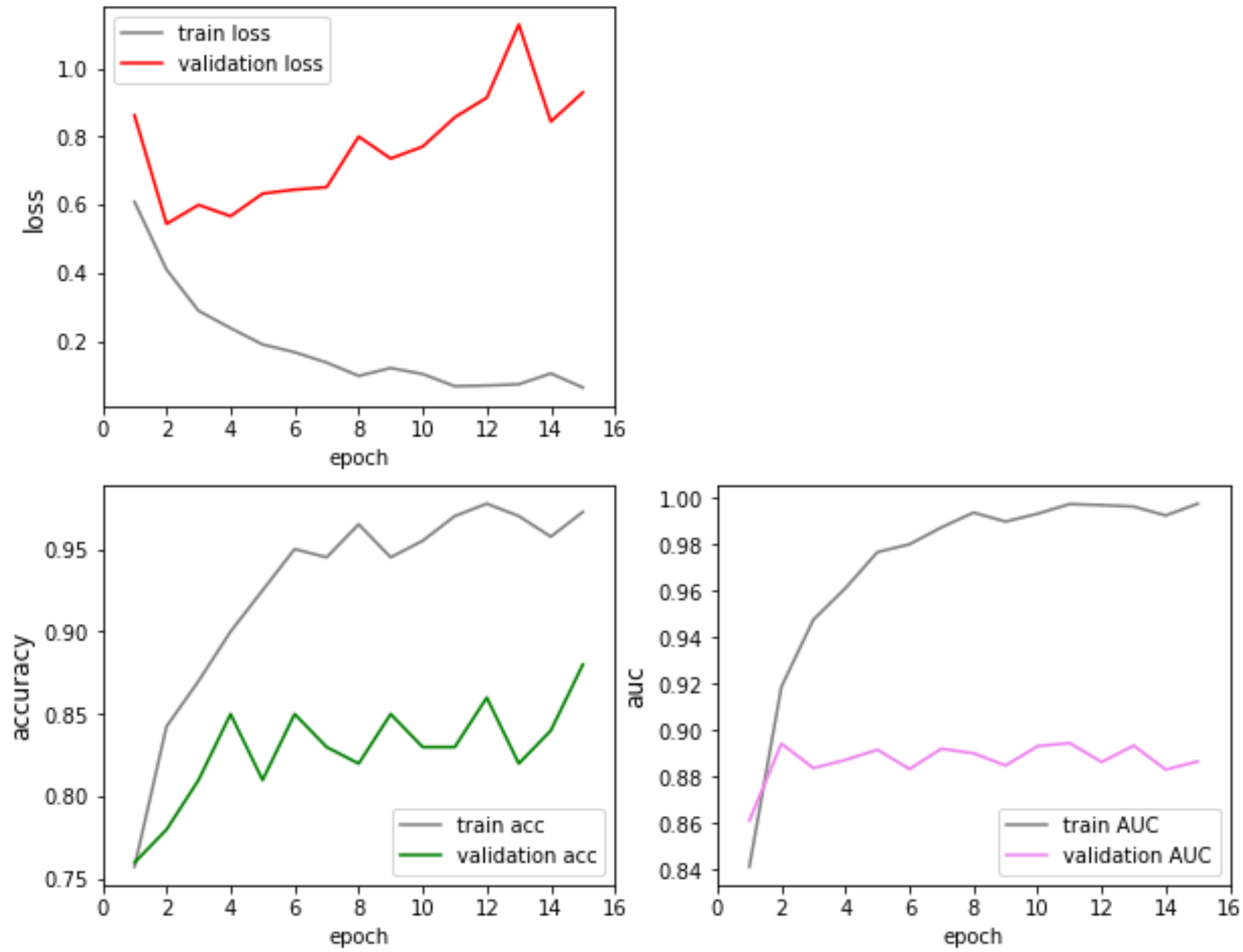


train loss
validation loss
loss
epoch
train acc
validation acc
accuracy
epoch
train AUC
validation AUC
auc
epoch

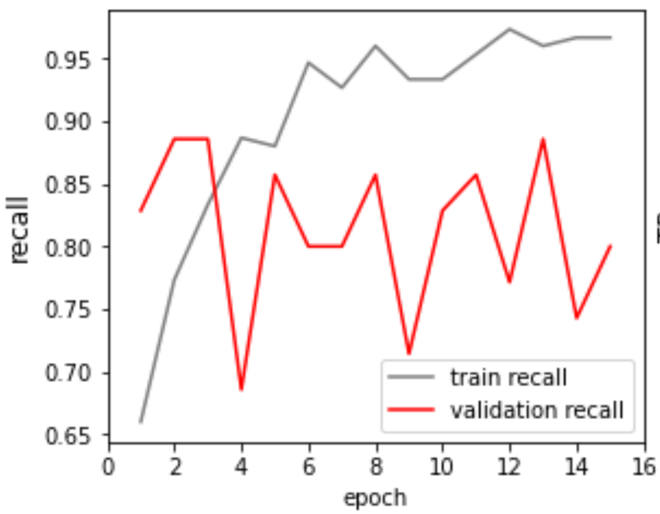

recall
epoch
train recall
validation recall
TP
train TP
validation TP
TN
train TN
validation TN
FP
train FP
validation FP
FN
train FN
validation FN

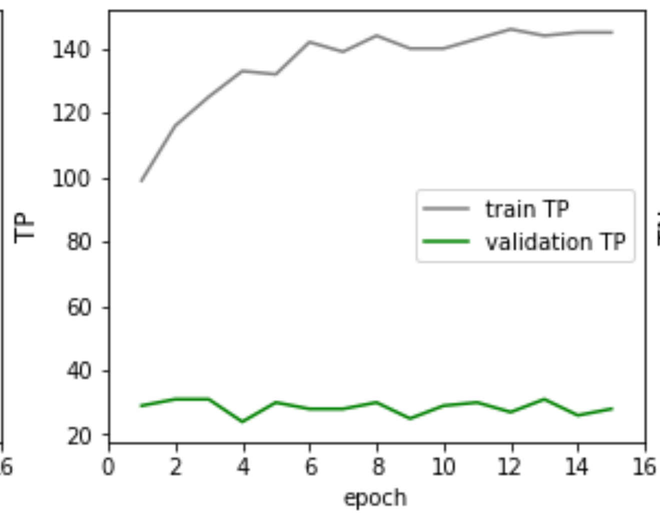
Model 2

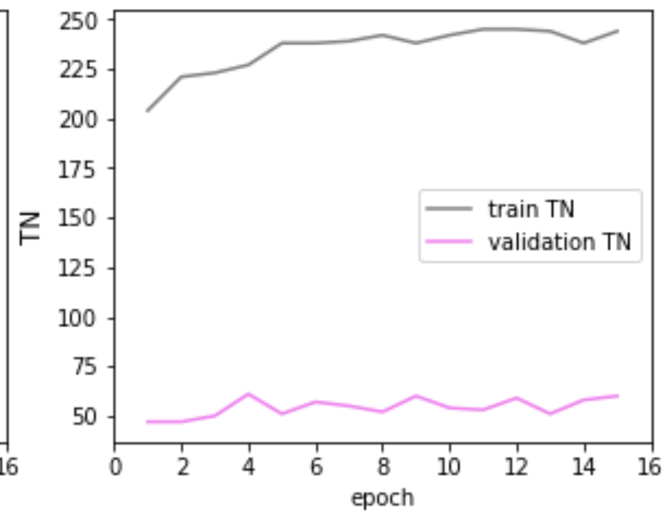
Model 3



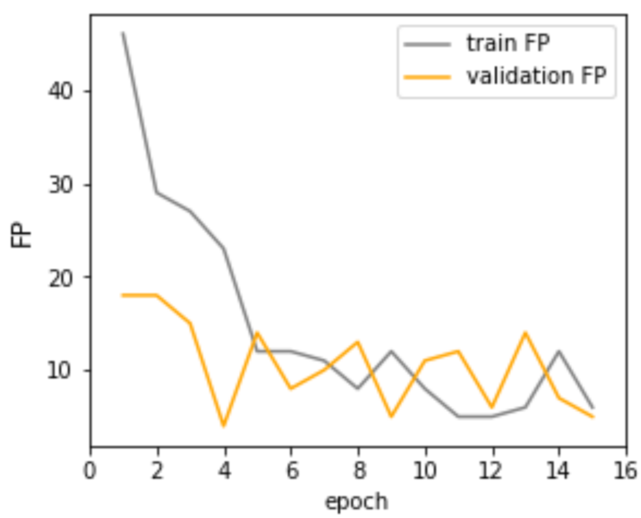
train loss
validation loss
loss
epoch
train acc
validation acc
accuracy
epoch
train AUC
validation AUC
auc
epoch

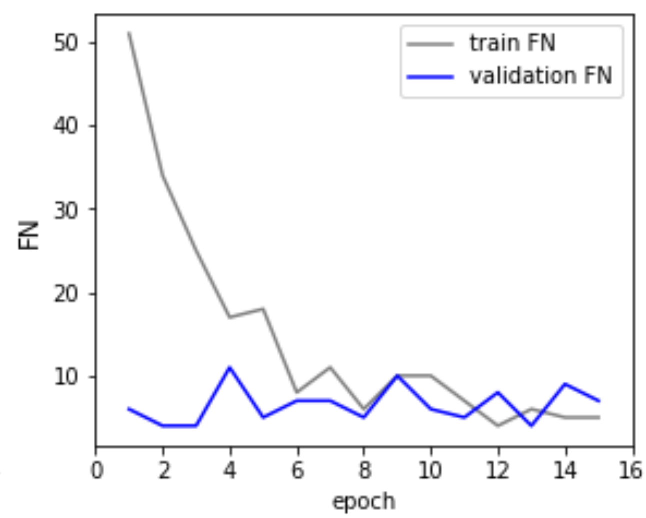

recall
epoch
train recall
validation recall


TP
epoch
train TP
validation TP


TN
epoch
train TN
validation TN

Model 3


FP
epoch
train FP
validation FP


FN
epoch
train FN
validation FN

Model 1

Train (Duke 400)

	TP	TN	FP	FN	sens	spec
0	106.0	192.0	58.0	44.0	0.706667	0.768
1	128.0	234.0	16.0	22.0	0.853333	0.936
2	139.0	233.0	17.0	11.0	0.926667	0.932
3	145.0	239.0	11.0	5.0	0.966667	0.956
4	146.0	240.0	10.0	4.0	0.973333	0.960
5	144.0	242.0	8.0	6.0	0.960000	0.968
6	148.0	247.0	3.0	2.0	0.986667	0.988
7	148.0	248.0	2.0	2.0	0.986667	0.992
8	148.0	244.0	6.0	2.0	0.986667	0.976
9	149.0	248.0	2.0	1.0	0.993333	0.992
10	150.0	250.0	0.0	0.0	1.000000	1.000
11	150.0	250.0	0.0	0.0	1.000000	1.000
12	150.0	250.0	0.0	0.0	1.000000	1.000
13	150.0	250.0	0.0	0.0	1.000000	1.000
14	150.0	250.0	0.0	0.0	1.000000	1.000

Validation (Duke 100)

	TP_val	TN_val	FP_val	FN_val	sens_val	spec_val
0	22.0	60.0	5.0	13.0	0.628571	0.923077
1	26.0	51.0	14.0	9.0	0.742857	0.784615
2	27.0	54.0	11.0	8.0	0.771429	0.830769
3	26.0	55.0	10.0	9.0	0.742857	0.846154
4	30.0	53.0	12.0	5.0	0.857143	0.815385
5	21.0	60.0	5.0	14.0	0.600000	0.923077
6	26.0	54.0	11.0	9.0	0.742857	0.830769
7	25.0	56.0	9.0	10.0	0.714286	0.861538
8	26.0	55.0	10.0	9.0	0.742857	0.846154
9	25.0	58.0	7.0	10.0	0.714286	0.892308
10	29.0	53.0	12.0	6.0	0.828571	0.815385
11	29.0	55.0	10.0	6.0	0.828571	0.846154
12	29.0	52.0	13.0	6.0	0.828571	0.800000
13	29.0	53.0	12.0	6.0	0.828571	0.815385
14	25.0	57.0	8.0	10.0	0.714286	0.876923

sensitivity, **recall**, **hit rate**, or **true positive rate** (TPR)

$$\mathrm{TPR} = \frac{\mathrm{TP}}{\mathrm{P}} = \frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FN}} = 1 - \mathrm{FNR}$$

specificity, **selectivity** or **true negative rate** (TNR)

$$\mathrm{TNR} = \frac{\mathrm{TN}}{\mathrm{N}} = \frac{\mathrm{TN}}{\mathrm{TN}+\mathrm{FP}} = 1 - \mathrm{FPR}$$

Validation:

- Specificity was OK.
- Sensitivity was bad

Solution:

- increase power
- Increase sample size

Test: Method 3 – Interpolating (on UAB)

	validation (M1)	test (M1)	validation (M2)	test (M2)	validation (M3)	test (M3)
loss	0.446491	0.327347	0.695937	0.362315	0.928706	0.707755
accuracy	0.830000	0.847176	0.860000	0.860465	0.880000	0.807309
auc	0.896703	0.914109	0.885055	0.912788	0.886593	0.908269
recall	0.657143	0.869565	0.685714	0.891304	0.800000	0.934783
TP	23.000000	40.000000	24.000000	41.000000	28.000000	43.000000
TN	60.000000	215.000000	62.000000	218.000000	60.000000	200.000000
FP	5.000000	40.000000	3.000000	37.000000	5.000000	55.000000
FN	12.000000	6.000000	11.000000	5.000000	7.000000	3.000000

Validation on Duke (100)
Test on UAB (301)

```
model1.save("VGG1_interpolating.h5")
model2.save("VGG2_interpolating.h5")
model3.save("VGG3_interpolating.h5")
```

Data backup

4 Folders:

- Competitor Company_Yuhui 20220819
- Duke project_python_Yuhui 20220819
- UAB project_python_Yuhui 20220819
- Zeiss PPT - work report_Yuhui 20220819

3 locations:

- Network
- OneDrive
- Remote computer (desktop)

Future

1. Upsampling:

- Pad out 10X10 to 224X224 (with 0 or NaN) (done (1))
- Upsample smaller image (Keras) (done (3))
- standard interpolator (scipy.interpolate.griddata — SciPy v1.9.0 Manual) (done (4))

2. Fine-tuning the transfer study of VGG16 (done (2))

- Hyperparameter tuning: AutoML