MACHINE LEARNING MODEL

COMPARISON BASED ON SOME METRICS

Safa ORHAN

Computer Engineering Student

Istanbul Kultur University

Istanbul, Turkey

Eyüp USTA

Computer Engineering Student

Istanbul Kultur University

Istanbul, Turkey

I. PAPER IMPLEMENTATION

1. Comparison of Classification Algorithms for Detection of Phishing Websites

K-Nearest Neighbors: # of neighbors = 5, weights = uniform, algorithm = auto

Paper Accuracy: 94.81% --- Our Accuracy: 93.72%

Multilayer Perceptron: Hidden Layers = 30, max iterations = 3000

Paper Accuracy: 97.22% --- Our Accuracy: 95.41%

Multilayer Perceptron: Hidden Layers = 150, max iterations = 1000

Paper Accuracy: 90.28% --- Our Accuracy: 96.98%

Multilayer Perceptron: Hidden Layers = 100, max iterations = 1000

Paper Accuracy: 96.71% --- Our Accuracy: 97.04%

Random Forest: # of estimators = 7, max depth = 11, criteria = entropy

Paper Accuracy: 95.25% --- Our Accuracy: 95.17%

Random Forest: # of estimators = 7, max depth = 8, criteria = entropy

Paper Accuracy: 89.16% --- Our Accuracy: 95.38%

SVC: C = 1.0, kernel = linear

Paper Accuracy: 92.71% --- Our Accuracy: 92.67%

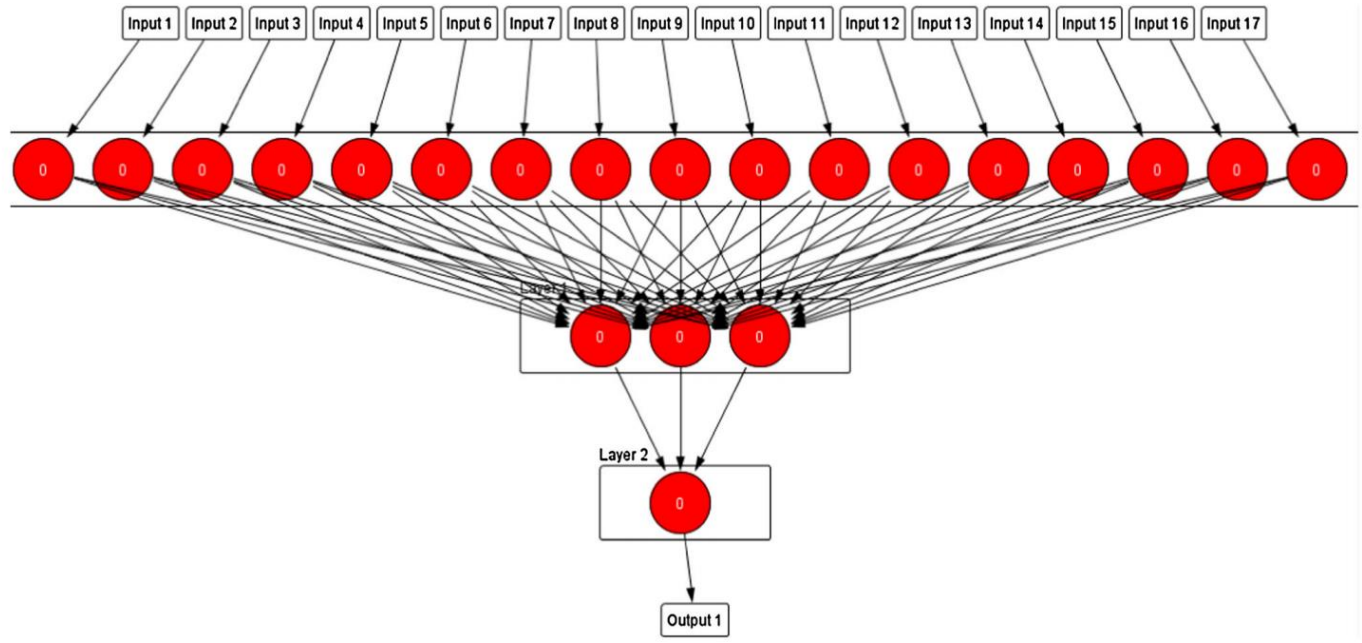
SVC: C = 1.0, kernel = polynomial, degree = 1

Paper Accuracy: 92.57% --- Our Accuracy: 92.58%

SVC: C = 1.0, kernel = polynomial, degree = 2

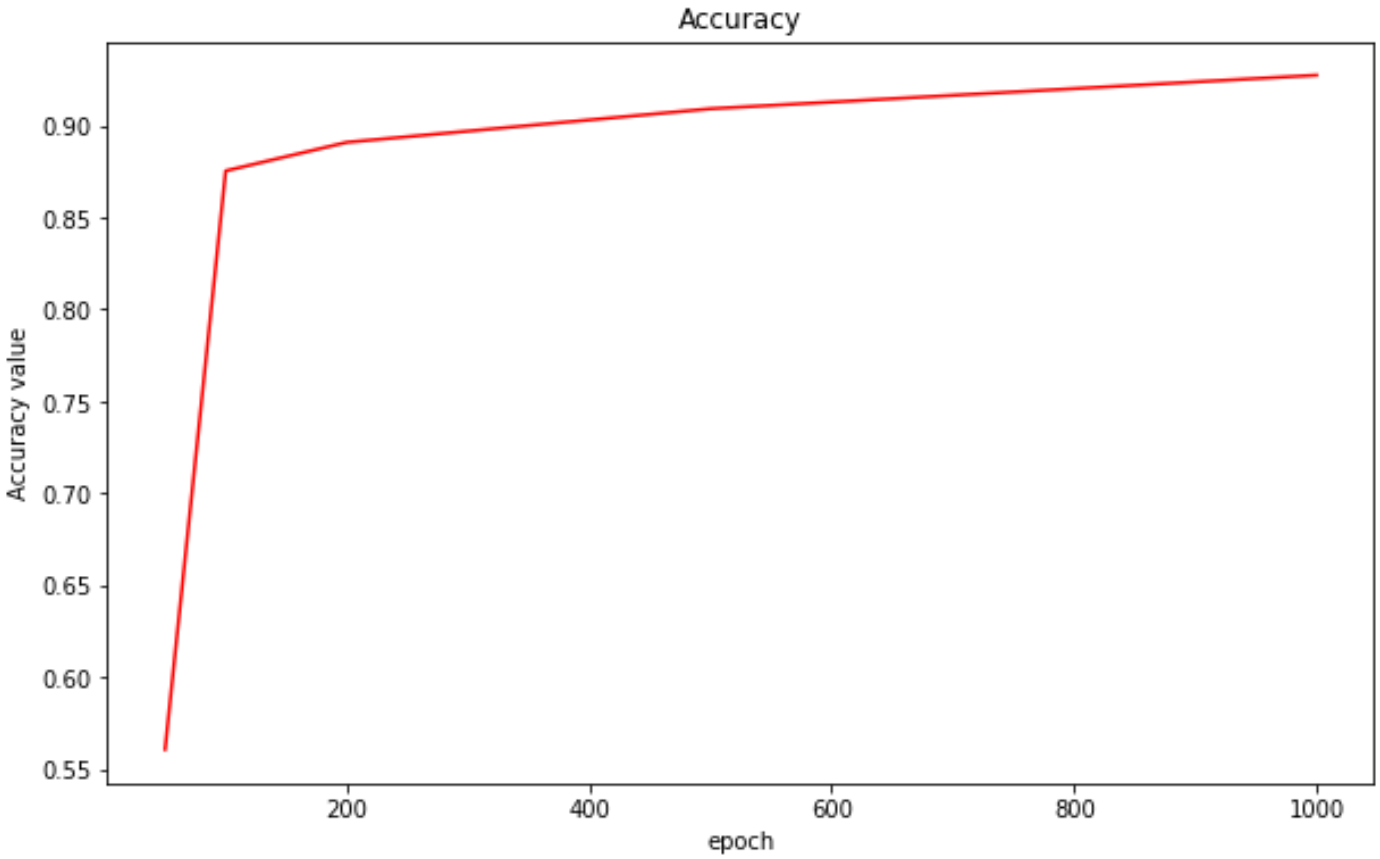
Paper Accuracy: 93.88% --- Our Accuracy 94.21%

2. Predicting phishing websites based on self-structuring neural network

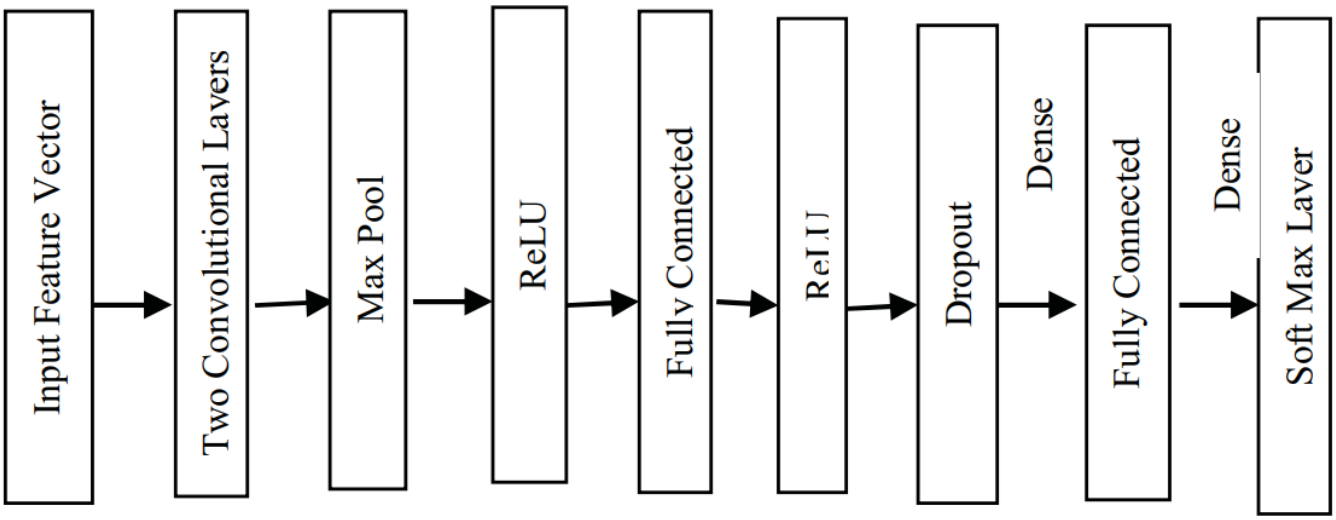


Features = having_IP_Address', 'URL_Length', 'having_At_Symbol', 'Prefix_Suffix', 'Abnormal_URL', 'SFH', 'HTTPS_token', 'Iframe', 'RightClick', 'popUpWidnow', 'having_Sub_Domain', 'Request_URL', 'URL_of_Anchor', 'Redirect', 'age_of_domain', 'DNSRecord', 'web_traffic', 'Result'

	Ours	Papers
Epoch: 50	**56.09%**	**91.32%**
Epoch: 100	**87.52%**	**92.33%**
Epoch: 200	**89.07%**	**93.07%**
Epoch: 500	**90.90%**	**93.45%**
Epoch: 1000	**92.72%**	**94.07%**



3. Automated Prediction of Phishing Websites Using Deep Convolutional Neural Network



Accuracy of Paper: 99.3%

Accuracy that we found: 55.43%

4. Phishing Website Detection Using Effective Classifiers and Feature Selection Techniques

Category	Feature Name	Value
Address bar	Having IP Address	1,0
	Having long url	1,0,-1
	Uses ShortningService	0,1
	Having '@' Symbol	0,1
	Double slash redirecting	0,1
	Having Prefix Suffix	-1,0,1
	Having Sub Domain	-1,0,1
	SSLfinal State	-1,1,0
	Domain registration Length	0,1,-1
	Favicon	0,1
	Is standard Port	0,1
	Uses HTTPS token	0,1
Abnormality	Request URL	1,-1
	Abnormal URL anchor	-1,0,1
	Links in tags	1,-1,0
	SFH	-1,1
	Submitting to email	1,-1
	Abnormal URL	1,0
HTML-JavaScript	Redirect	0,1
	on mouseover	0,1
	RightClick	1,-1
	popUpWindow	-1,1
	Iframe	0,1
Domain	Age of domain	-1,0,1
	on DNS Record	1,0
	Web traffic	-1,0,1
	Page Rank	-1,0,1
	Google Index	0,1
	Links pointing to page	1,0,-1
	Statistical report	1,0

PERFORMANCE OF CLASSIFIERS FOR ADDRESS BAR BASED FEATURES ONLY

Naive Bayes: 89.59% Paper: 89.95%

Decision Tree: 90.32% Paper: 90.19%

PERFORMANCE OF CLASSIFIERS FOR ABNORMAL BASED FEATURES ONLY

Naive Bayes: 72.20% Paper: 88.45%

Decision Tree: 87.63% Paper: 89.05%

PERFORMANCE OF CLASSIFIERS FOR JAVASCRIPT and HTML BASED FEATURES ONLY

Naive Bayes: 56.01% Paper: 54.12%

Decision Tree: 57.43% Paper: 58.02%

PERFORMANCE OF CLASSIFIERS FOR DOMAIN BAR BASED FEATURES ONLY

Naive Bayes: 69.88% Paper: 80.35%

Decision Tree: 72.74% Paper: 81.55%

Voting Classifier

Models used:

Support Vector Machine, K-Nearest Neighbors, BNaive Bayes, Random Forest, Decision Tree

Accuracy = 96.89

II. DEEP LEARNING WITH TENSORFLOW

Optimizers = 'sgd', 'rmsprop', 'adam', 'adadelta', 'adagrad', 'adamax', 'nadam', 'ftrl'

Loss = 'binary_crossentropy', 'categorical_crossentropy', 'hinge', 'squared_hinge', 'huber'

Activation = 'softplus', 'softsign', 'selu', 'elu', 'exponential', 'tanh', 'sigmoid', 'relu'

A. Results of the Deep Network:

```
Flatten(input_shape = (17,)
Dense(32, kernel_regularizer=l2(0.0001), activation='sigmoid')
Dense(64, activation='sigmoid')
Dense(32,activation='sigmoid')
Dense(1,activation='sigmoid')
loss='binary_crossentropy', optimizer='sgd', metrics=['accuracy'], epochs = 1000
```

Accuracy = 92.82%

Categorical Features:

```
Flatten(input_shape = (17,2)
Dense(32, kernel_regularizer=l2(0.0001), activation='sigmoid')
Dense(64, activation='sigmoid')
Dense(32,activation='sigmoid')
Dense(1,activation='sigmoid')
loss='binary_crossentropy', optimizer='sgd', metrics=['accuracy'], epochs = 1000
```

Accuracy = 72.02%

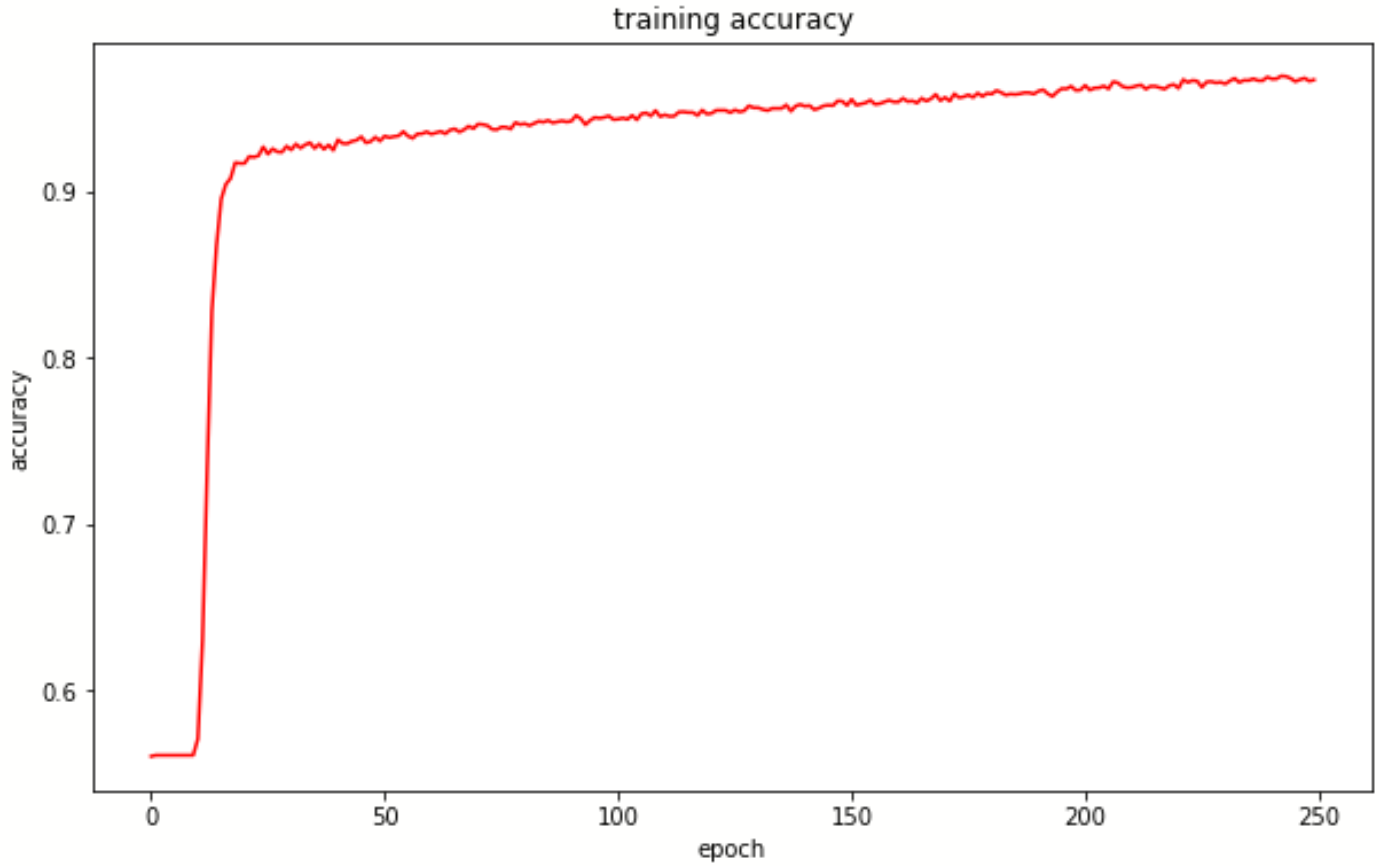
B. Recurrent Neural Networks:

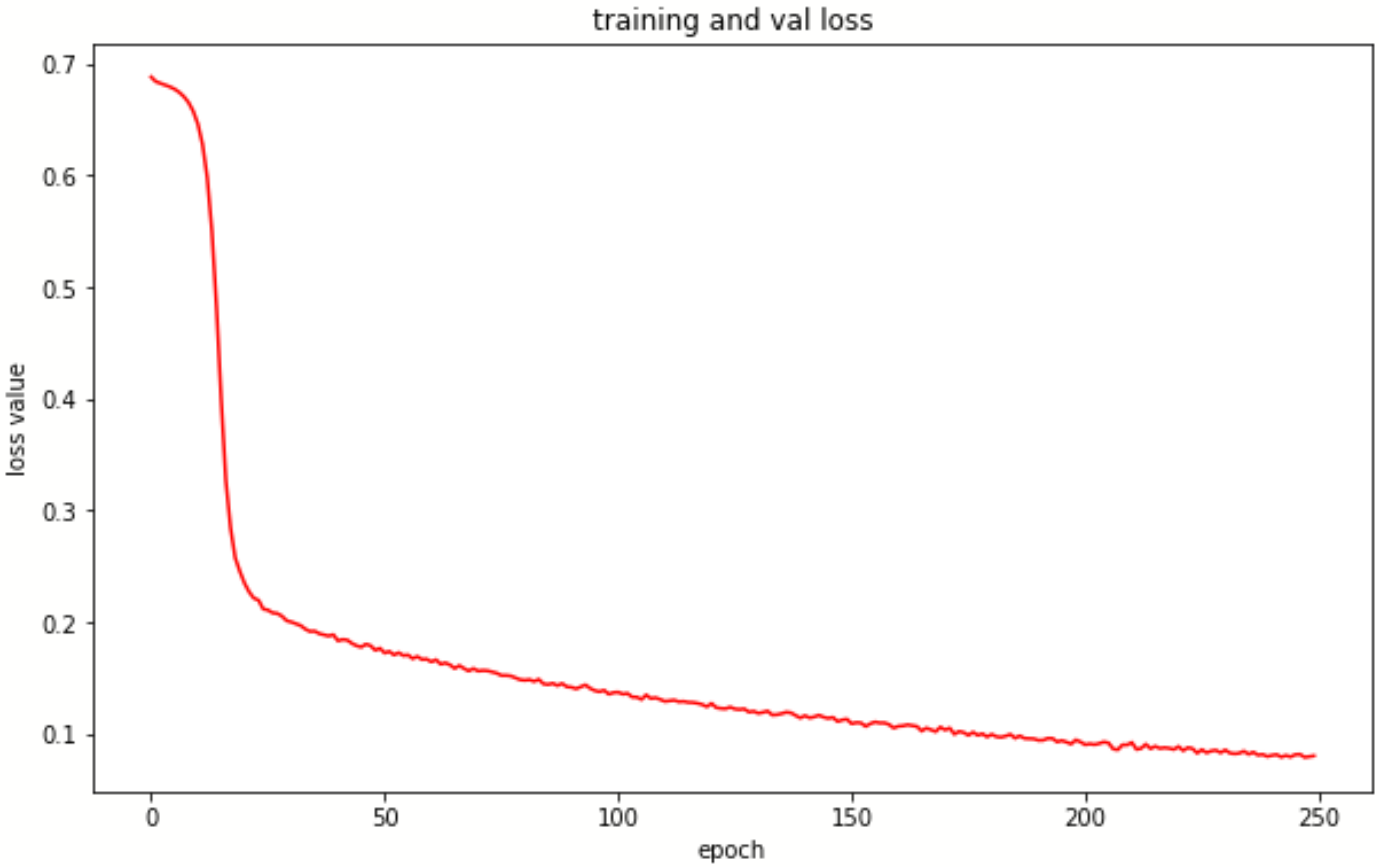
Model 1:

```
keras.layers.LSTM(128, activation='relu',return_sequences=True, input_shape=(1,30))
keras.layers.Dropout(0.2)
keras.layers.LSTM(256, activation='relu')
keras.layers.Dropout(0.2)
keras.layers.Dense(250,activation='relu')
keras.layers.Dense(1,activation='sigmoid')
```

Accuracy = 97.17%

```
loss='binary_crossentropy', optimizer='sgd', metrics=['accuracy']
Epoch Number = 250
```


training accuracy
accuracy
0.9
0.8
0.7
0.6
0
50
100
150
200
250
epoch

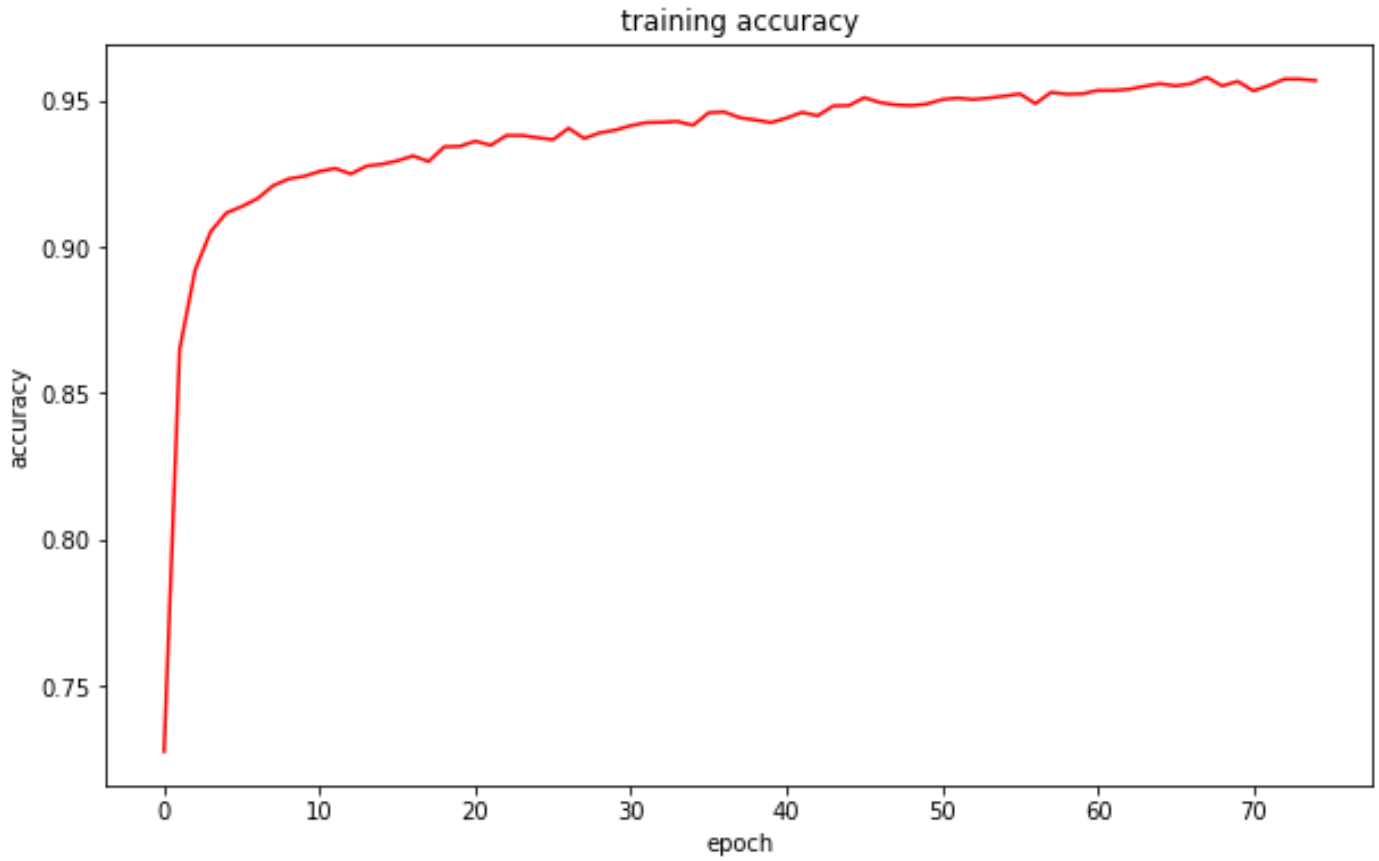

training and val loss
loss value
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
50
100
150
200
250
epoch

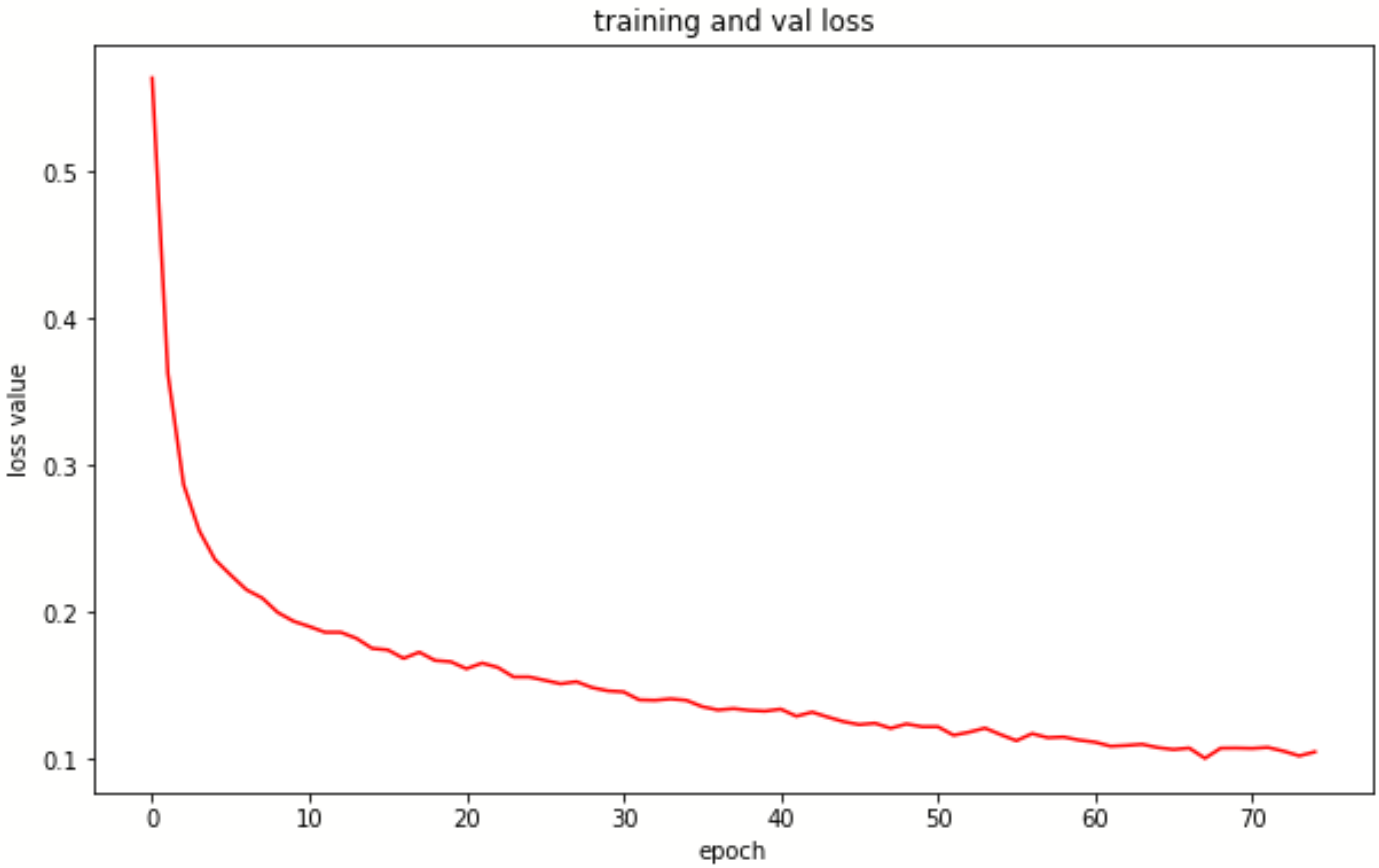
Model 2:

```
keras.layers.SimpleRNN(64, activation='relu',return_sequences=True, input_shape=(1,30))
keras.layers.Dropout(0.2)
keras.layers. SimpleRNN (128, activation='relu')
keras.layers.Dropout(0.2)
keras.layers.Dense(128,activation='relu')
keras.layers.Dense(128,activation='relu')
keras.layers.Dense(1,activation='sigmoid')
loss='binary_crossentropy', optimizer='sgd', metrics=['accuracy']
```

Accuracy = 96.54%

Epoch Number = 75





Model 3:

keras.layers.GRU(64, activation='relu',return_sequences=True, input_shape=(1,30))

keras.layers.Dropout(0.2)

keras.layers. SimpleRNN (128, activation='relu')

keras.layers.Dropout(0.2)

keras.layers.Dense(128,activation='relu')

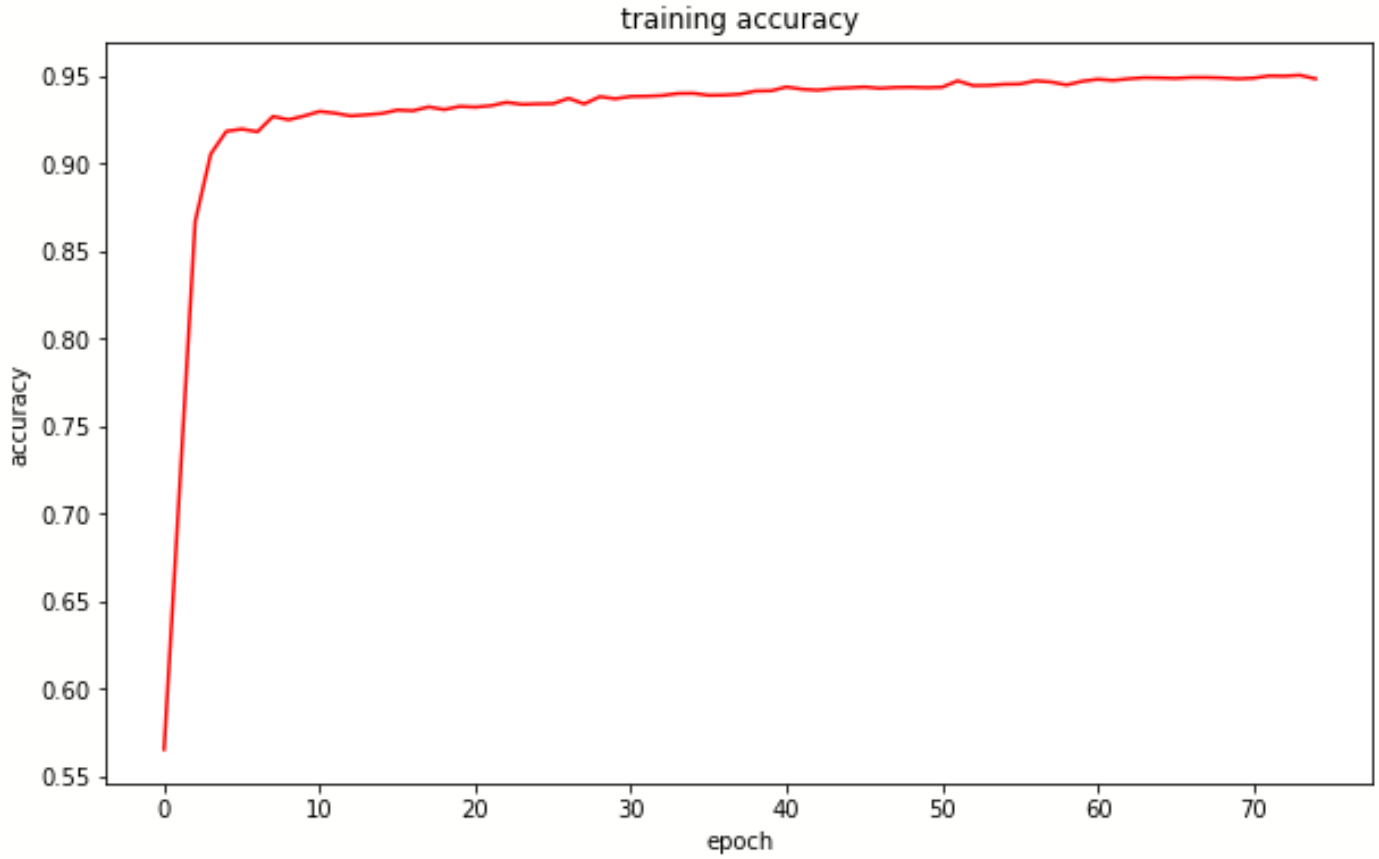
keras.layers.Dense(128,activation='relu')

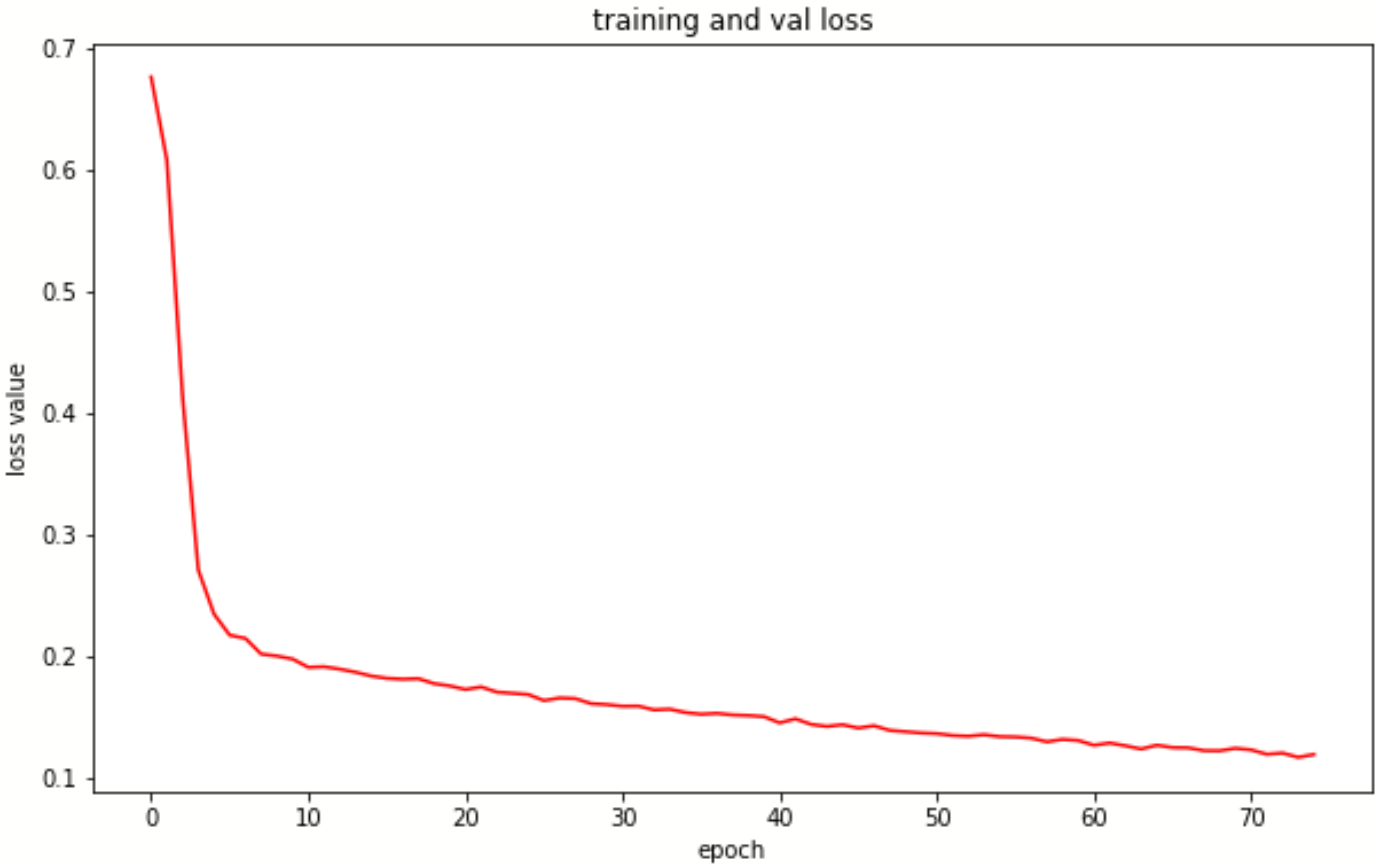
keras.layers.Dense(1,activation='sigmoid')

loss='binary_crossentropy', optimizer='sgd', metrics=['accuracy']

Accuracy = 95.11%

Epoch Number = 75


training accuracy
accuracy
epoch
0.95
0.90
0.85
0.80
0.75
0.70
0.65
0.60
0.55
0
10
20
30
40
50
60
70


training and val loss
loss value
epoch
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
10
20
30
40
50
60
70