Final Presentation

July 10, 2017

Prerit Gupta
Summer Intern

Ravi Kiran Sarvadevabhatla
PhD Student

Objective

- Visualization and Understanding of different CNN architectures : AlexNet, VGG-19, GoogLeNet, ResNet for sketch classification
- Understanding the Alexnet + GRU and Alexnet + LSTM architecture for classifying sketches utilizing the sequential stroke order.
- Comparing CNN and RNN architectures

Roadmap

1. Reading papers, exploring different visualization techniques in literature — May 3 - May 13
2. Getting familiarized with caffe models & files. — May 13 - May 18
3. Understanding the code of Yosinski's Deep Visualization toolbox — May 18 - May 22
4. Maximally activated images & deconvolution — May 23 - May 25
5. Running toolbox for Alexnet & plotting results — May 26 - May 28
6. Tiling results & plotting histograms to get top 5 & bottom 5 images — May 29 - June 1
7. Obtaining results for VGG-19 — June 2 - June 5
8. Obtaining results for alexnet trained with sketch dataset mean — June 6 - June 9
9. Theano & Lassagne tutorials , plotting first layer filter weights — June 10 - June 12
10. More analysis for the results for CNN, included googlenet — June 13 - June 20
11. Alexnet+GRU & Alexnet + LSTM Visualization — June 21 - June 30
12. Obtained hidden vectors, top3 & bot 3 activated classes — July 1 - July 9

Analysis through Deep Visualization Toolbox

- Software tool that provides a live, interactive visualization of every neuron in a trained convnet as it responds to a user-provided image or video.
- The tool displays forward activation values, preferred stimuli via gradient ascent, top images for each unit from the training set, deconv highlighting (Zeiler & Fergus, 2013) of top images, and backward diffs computed via backprop or deconv starting from arbitrary units.

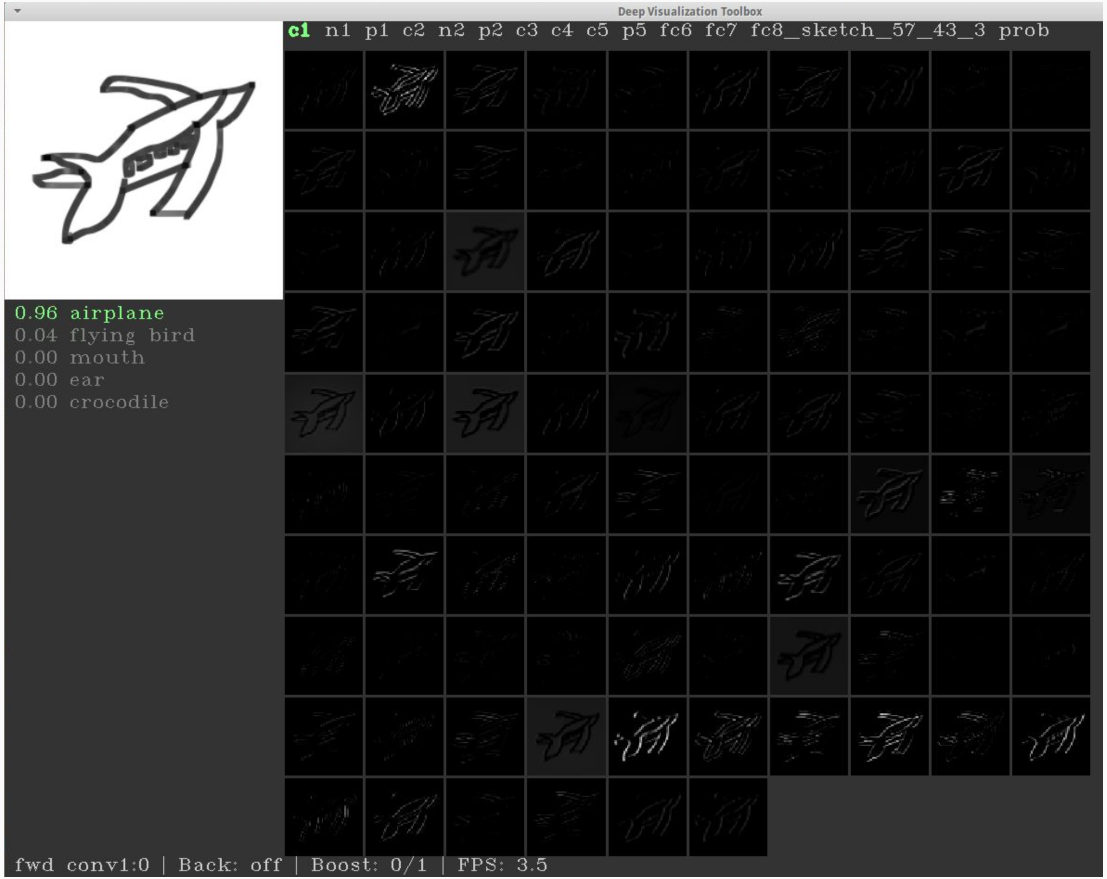
Reference: J. Yosinki, Understanding Neural Networks through Deep Visualization, ICML DL Workshop, 2015
Github Repository: https://github.com/yosinski/deep-visualization-toolbox
Link: http://yosinski.com/deepvis

Deep Visualization toolbox

Yosinki's Deep Visualization Toolbox is used to visualize different filters within selected layer for sketches.

Target Class: Airplane
Predicted Probabilities:

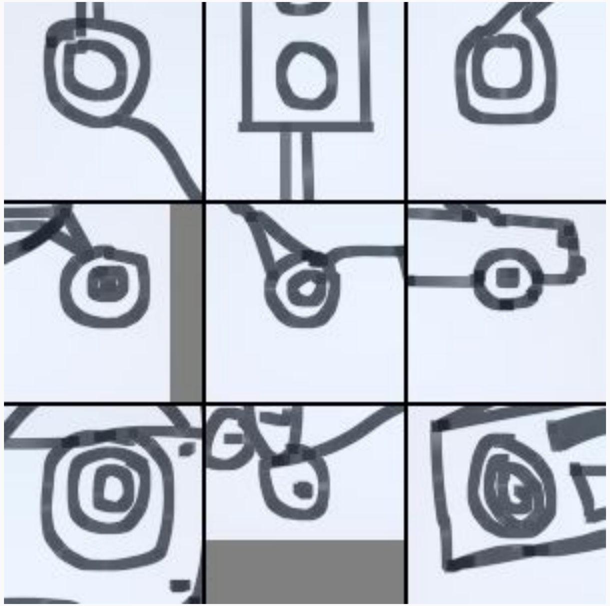
- 0.96 Airplane
- 0.04 Flying Bird



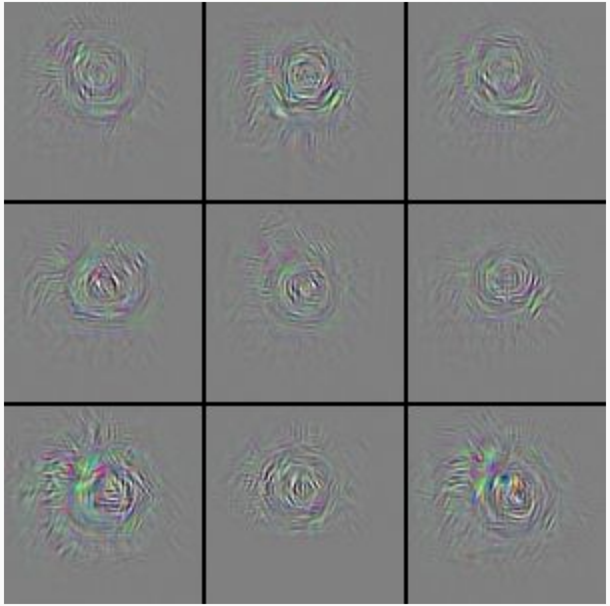
Visualizing layers by finding the Maximum Activated Input Patch & perform Deconvolution

- Finding the top 9 patches from the input images in validation set which maximally activate a filter for a certain layer.
- To understand which kind of features are learned by filters in different layers and observe their sensitivity towards certain classes & localized patches with certain geometry.
- Perform deconvolution for the filters that are activated by maximally activating input patches.

AlexNet Visualization : Convolution Layer 3



Input Patch for Maximum Activation



Deconvolution of the activated patch

- 99 x 99 Input Patch for 284th filter in 3rd convolutional layer.
- Maximum Activation: 428.343414
- Filter sensitive to specific parts of sketches having circles.
- Classes with performance: head-phones 100.0 traffic light 92.86 scissors 100.0 wheelbarrow 92.86 wheelbarrow 92.86 car (sedan) 92.86 camera 85.71 train 78.57 radio 57.14

Plotting histograms to understand class preference in each layer

- The top 9 maximally activated images for every filter in a layer are traced back to the classes which they belong.
- For every rank of activation across all filters in a layer, a histogram is plotted to represent the count of class a filter prefers.
- Hence, for every layer 9 histograms are obtained on whose weighted summation a single histogram is obtained for every layer.
- Weight of rank 1 class -> 9 , Rank 2 -> 8 & so on…. Rank 9 -> 1

Convolution Layer 5

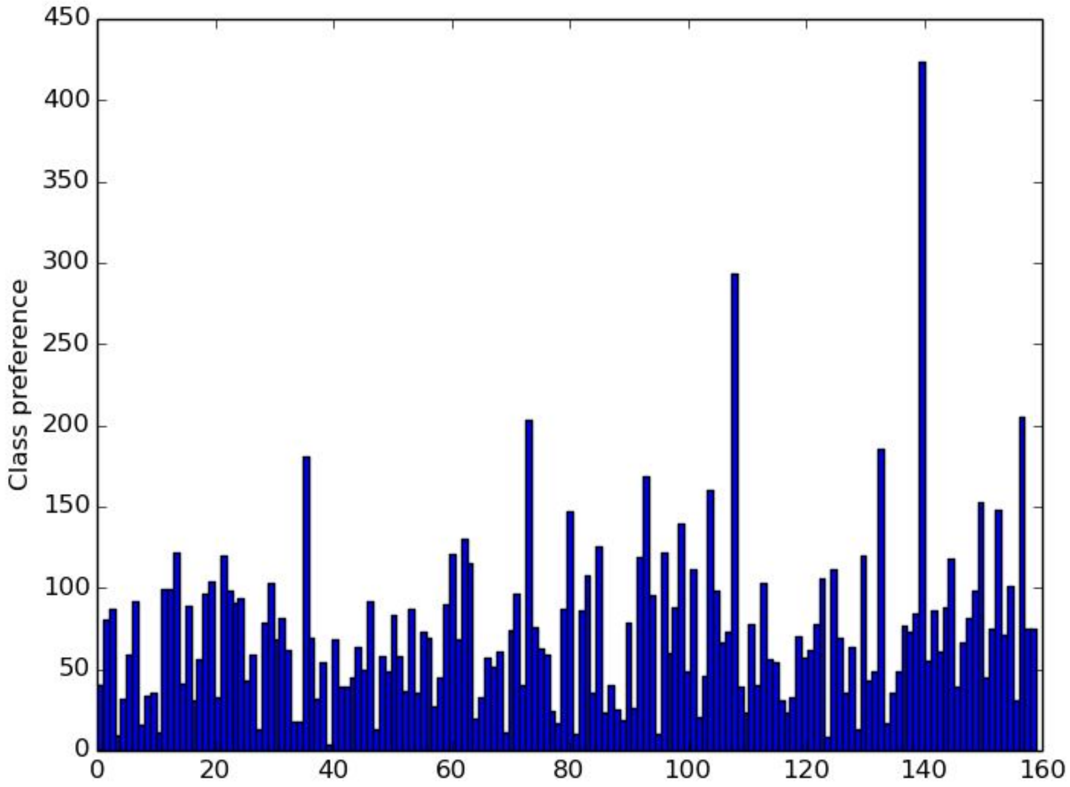
Inference

Top 5 prefered classes :

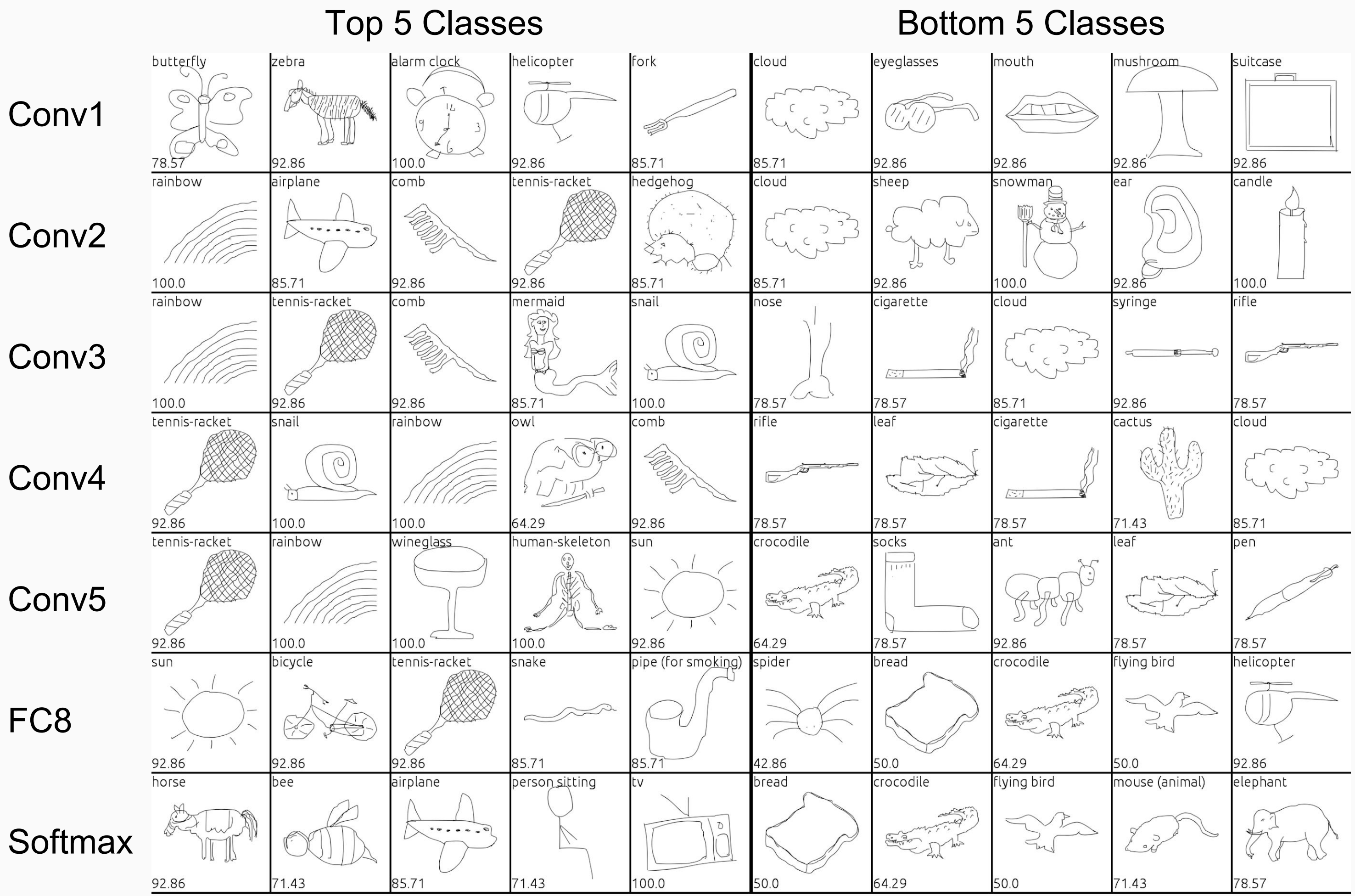
- Tennis-racket
- Sun
- Telephone
- Cake
- Rainbow

Least 5 prefered classes:

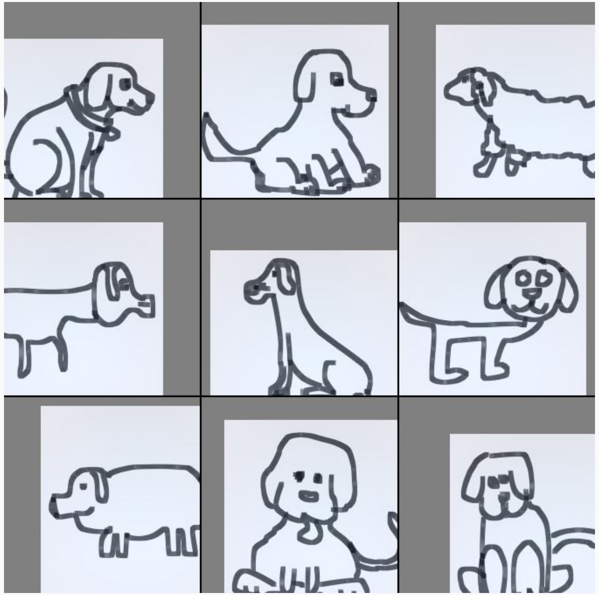
- Baseball bat
- Nose
- Rifle
- Bed
- Bowl



Summary Plot with class performance(Alexnet with Imagenet Mean)



VGG-19 : Conv5_4 Layer

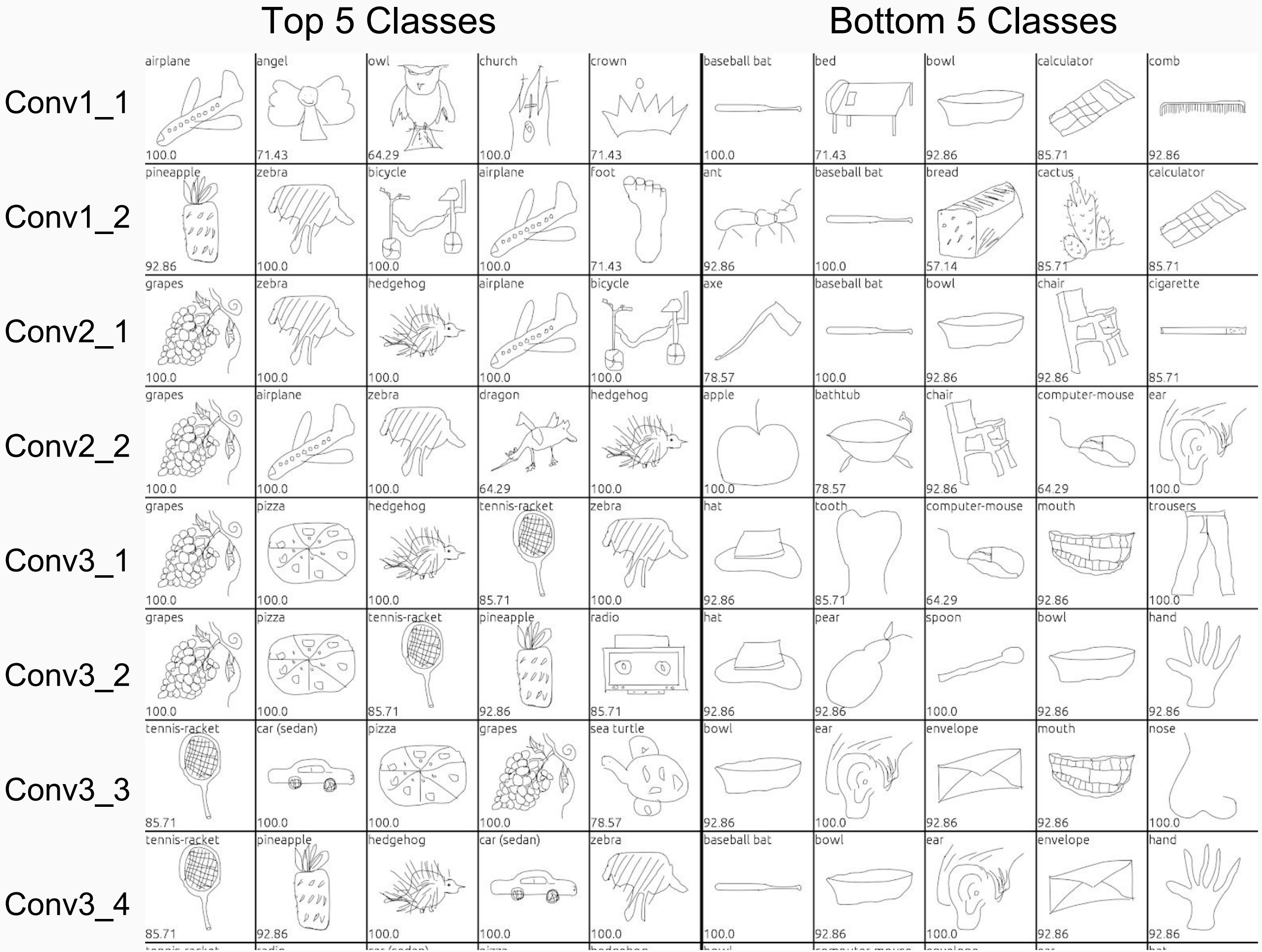


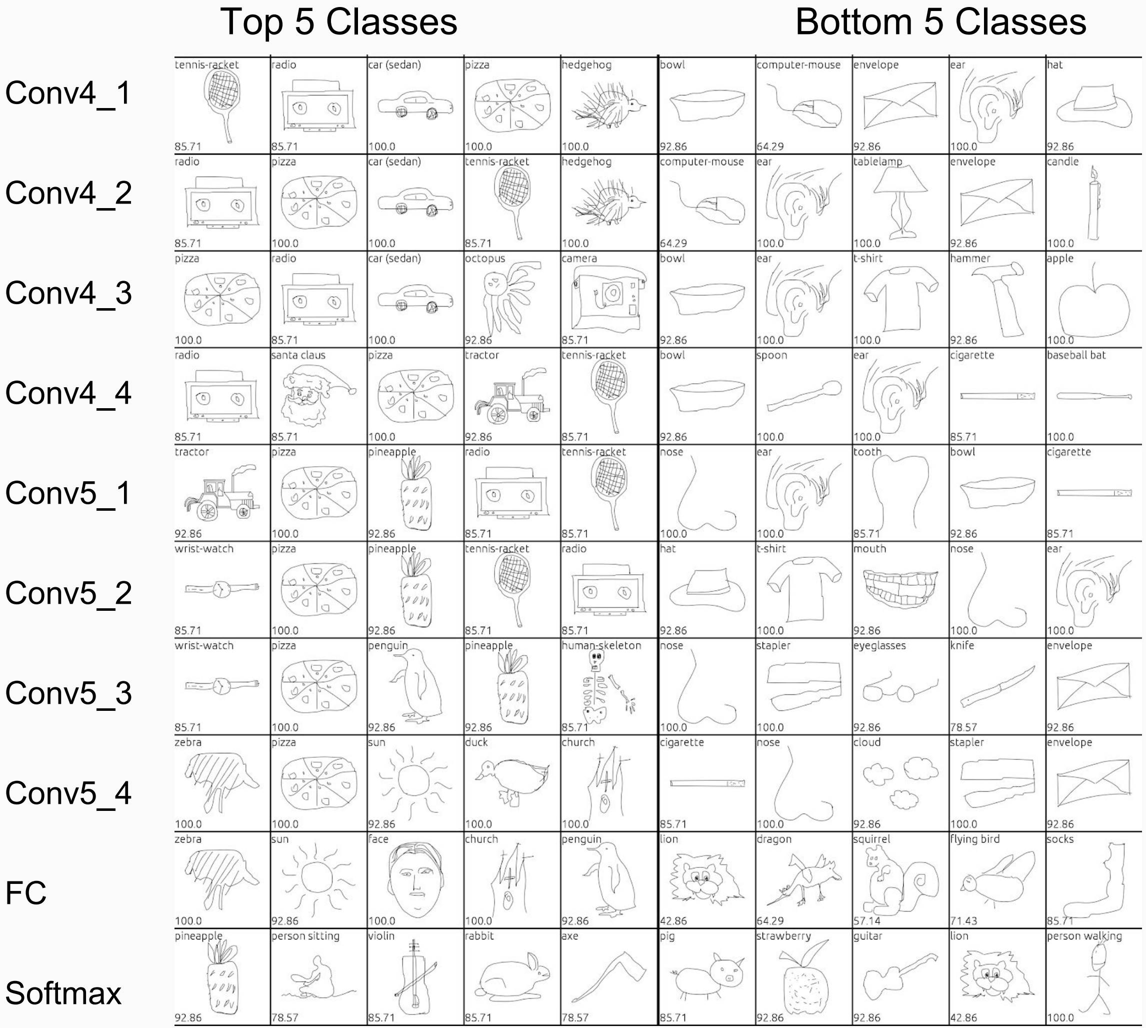
Input Patch for Maximum Activation

Deconvolution of the activated patch

- 224 x 224 Input Patch for 352nd filter in Conv5_4 layer
- Maximum Activation: 575.805237
- Filter sensitive to animals like dog, sheep, pig.
- Classes with performance: dog 50.0 dog 50.0 sheep 92.86 dog 50.0 dog 50.0 dog 50.0 pig 85.71 dog 50.0 dog 50.0

Summary Plot (VGG-19 with Imagenet Mean)





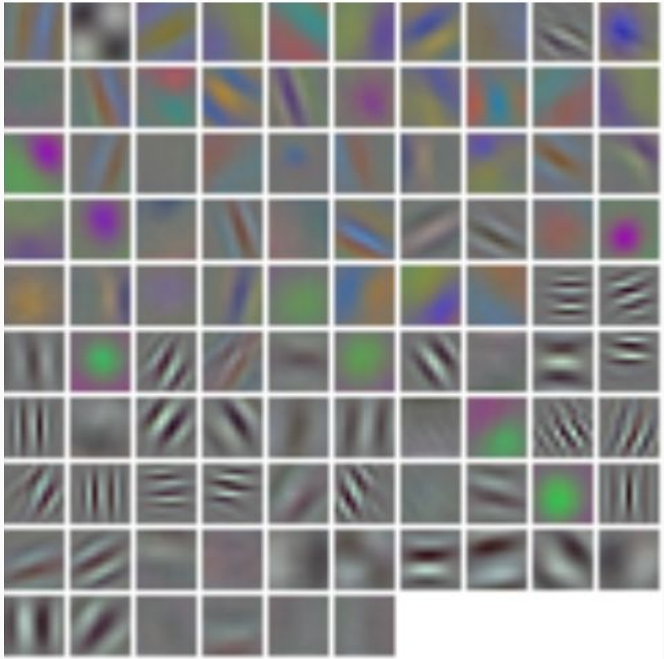
Top 5 Classes
Bottom 5 Classes
Conv4_1
tennis-racket 85.71
radio 85.71
car (sedan) 100.0
pizza 100.0
hedgehog 100.0
bowl 92.86
computer-mouse 64.29
envelope 92.86
ear 100.0
hat 92.86
Conv4_2
radio 85.71
pizza 100.0
car (sedan) 100.0
tennis-racket 85.71
hedgehog 100.0
computer-mouse 64.29
ear 100.0
tablelamp 100.0
envelope 92.86
candle 100.0
Conv4_3
pizza 100.0
radio 85.71
car (sedan) 100.0
octopus 92.86
camera 85.71
bowl 92.86
ear 100.0
t-shirt 100.0
hammer 92.86
apple 100.0
Conv4_4
radio 85.71
santa claus 85.71
pizza 100.0
tractor 92.86
tennis-racket 85.71
bowl 92.86
spoon 100.0
ear 100.0
cigarette 85.71
baseball bat 100.0
Conv5_1
tractor 92.86
pizza 100.0
pineapple 92.86
radio 85.71
tennis-racket 85.71
nose 100.0
ear 100.0
tooth 85.71
bowl 92.86
cigarette 85.71
Conv5_2
wrist-watch 85.71
pizza 100.0
pineapple 92.86
tennis-racket 85.71
radio 85.71
hat 92.86
t-shirt 100.0
mouth 92.86
nose 100.0
ear 100.0
Conv5_3
wrist-watch 85.71
pizza 100.0
penguin 92.86
pineapple 92.86
human-skeleton 85.71
nose 100.0
stapler 100.0
eyeglasses 92.86
knife 78.57
envelope 92.86
Conv5_4
zebra 100.0
pizza 100.0
sun 92.86
duck 100.0
church 100.0
cigarette 85.71
nose 100.0
cloud 92.86
stapler 100.0
envelope 92.86
FC
zebra 100.0
sun 92.86
face 100.0
church 100.0
penguin 92.86
lion 42.86
dragon 64.29
squirrel 57.14
flying bird 71.43
socks 85.71
Softmax
pineapple 92.86
person sitting 78.57
violin 85.71
rabbit 85.71
axe 78.57
pig 85.71
strawberry 92.86
guitar 92.86
lion 42.86
person walking 100.0

First Layer Filter Visualizations

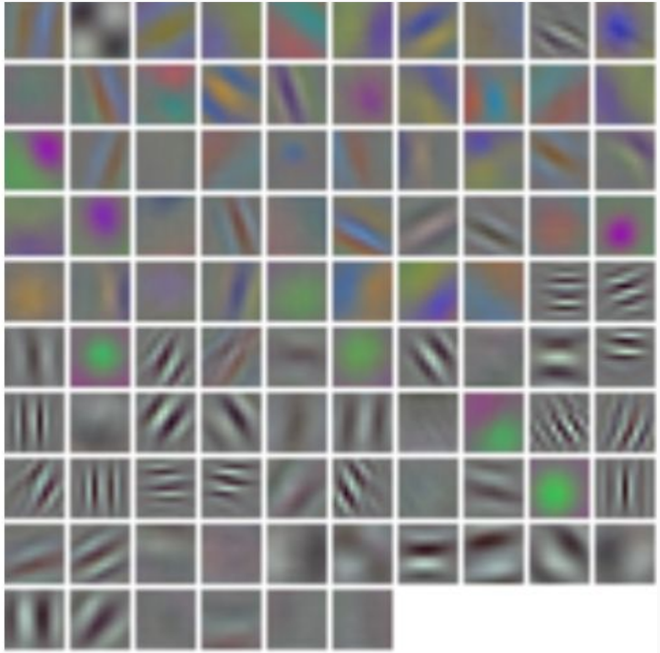
(AlexNet, GoogLeNet, VGG - 19, ResNet)

Code : http://nbviewer.jupyter.org/github/BVLC/caffe/blob/master/examples/00-classification.ipynb

Alexnet



Pretrained for images (Before fine-tuning)



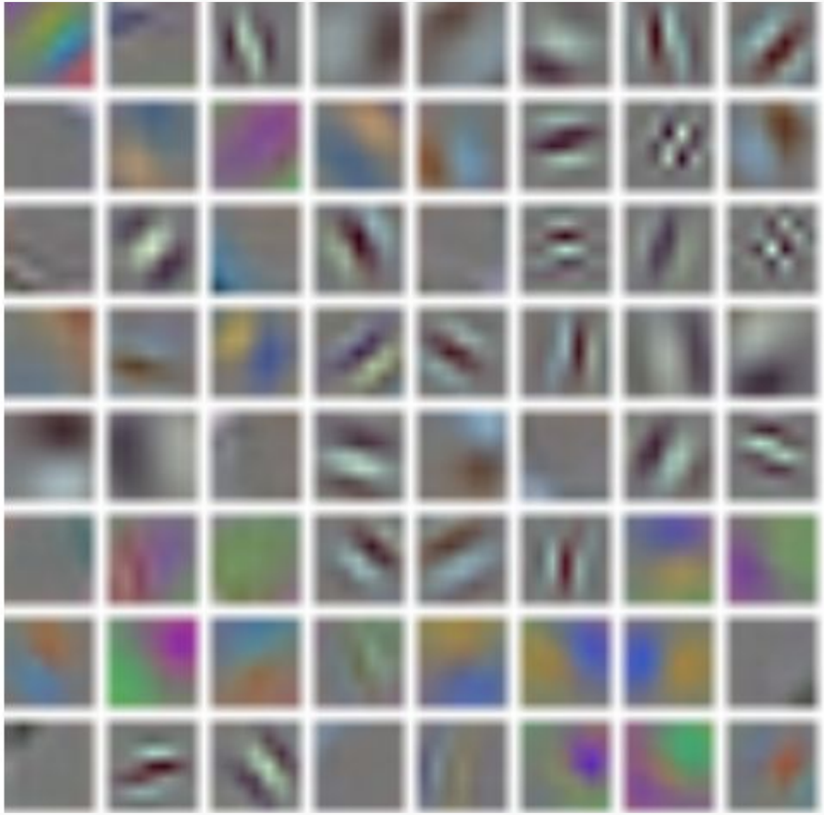
Trained for sketches (After fine-tuning)

VGG-19

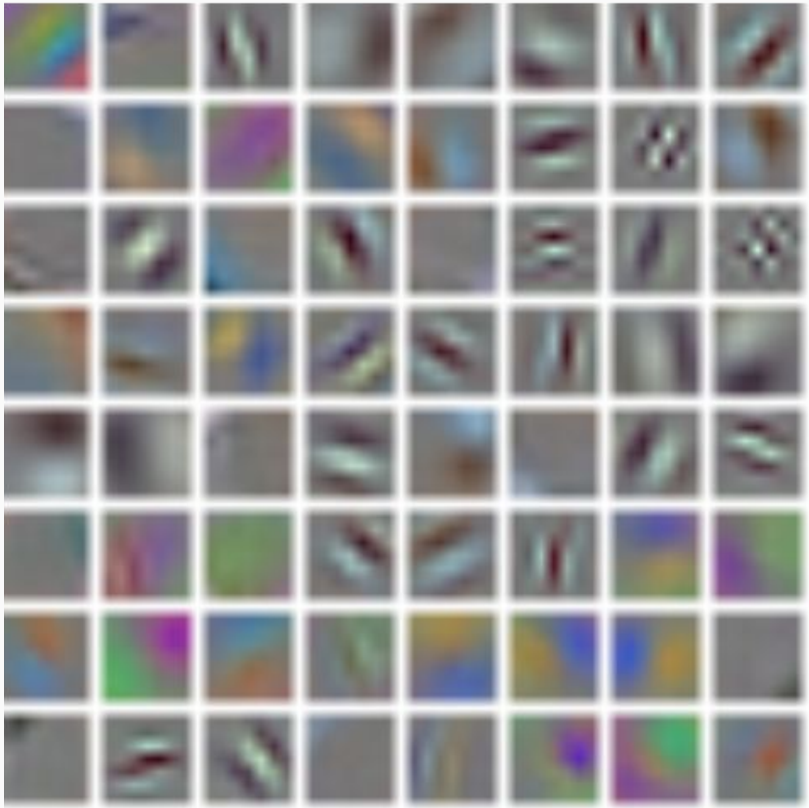
Pretrained for images (Before fine-tuning)

Trained for sketches (After fine-tuning)

GoogLeNet

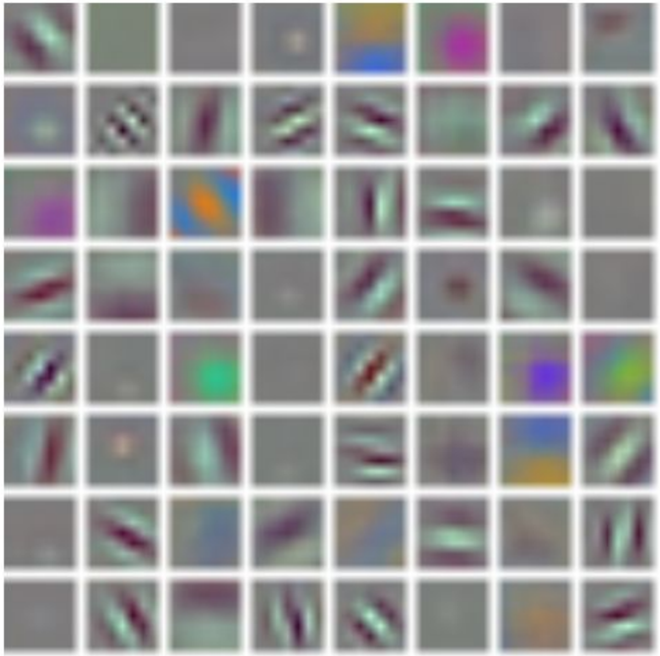


Pretrained for images (Before fine-tuning)



Trained for sketches (After fine-tuning)

ResNet-50



Trained for sketches (After fine-tuning)

Characterizing Visualizations in layers of GoogLeNet architecture

(Training of GoogLeNet uses mean file of ImageNet dataset)

conv1/7x7_s2

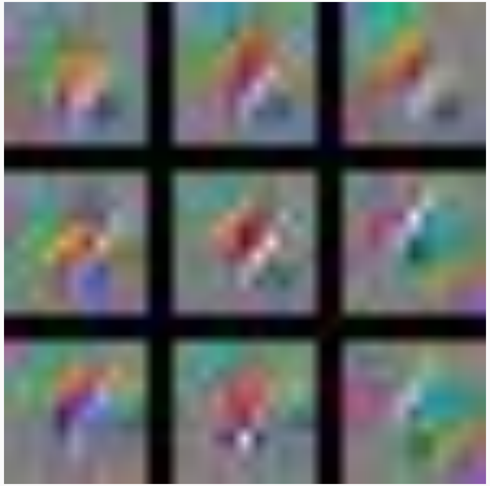
Input Patch for Maximum Activation

Deconvolution of the activated patch

- 33rd filter in 1st convolutional layer.
- Maximum activation : 3823.192627
- Filter sensitive to edge transition.
- Classes with performance: bicycle 100.0 baseball bat 78.57 trumpet 71.43 grapes 100.0 syringe 85.71 teapot 100.0 pear 100.0 bee 57.14 scorpion 71.43

conv2/3x3_reduce

Input Patch for Maximum Activation

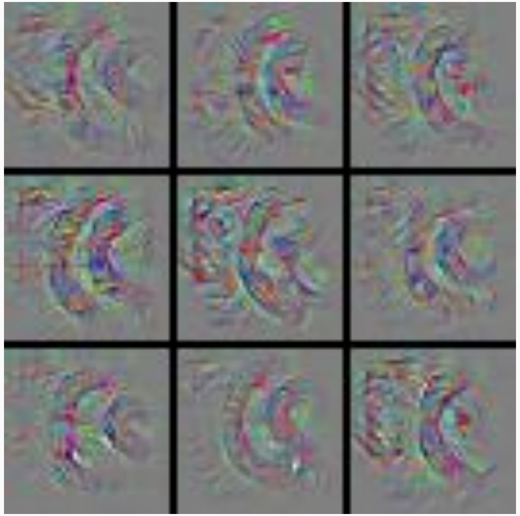


Deconvolution of the activated patch

- Input Patch for 43rd filter.
- Maximum activation : 345.881775
- Filter sensitive to edge transition.
- Classes with performance: person sitting 71.43 kangaroo 64.29 pizza 92.86 radio 85.71 rainbow 100.0 calculator 85.71 octopus 92.86 rabbit 78.57 ladder 100.0

inception_3a/3x3

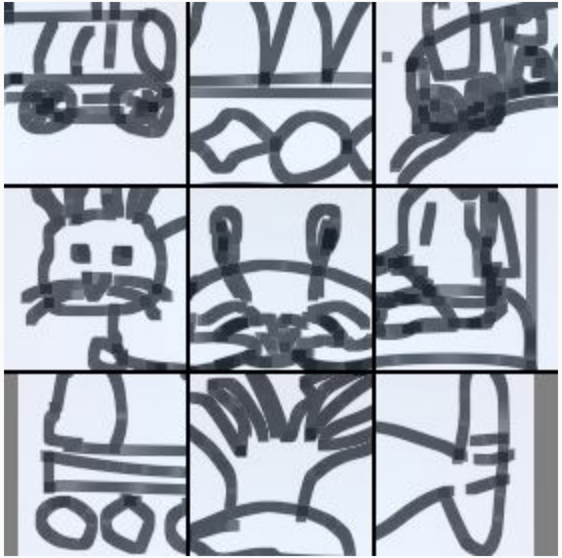
Input Patch for Maximum Activation



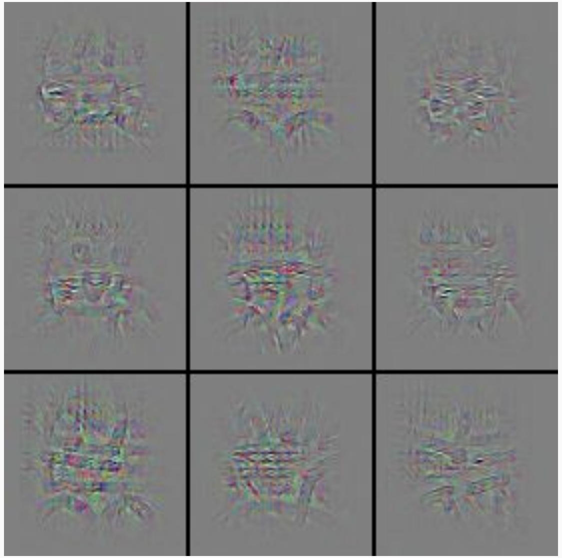
Deconvolution of the activated patch

- Input Patch for 2nd filter
- Maximum activation : 1011.357605
- Filter sensitive to curved patterns.
- Classes with performance: key 92.86 radio 85.71 tree 100.0 monkey 57.14 crab 78.57 monkey 57.14 shovel 92.86 mermaid 71.43 tree 100.0

inception_3b/5x5



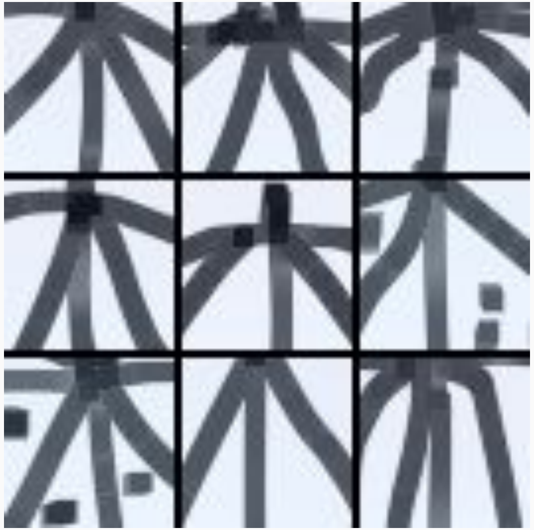
Input Patch for Maximum Activation



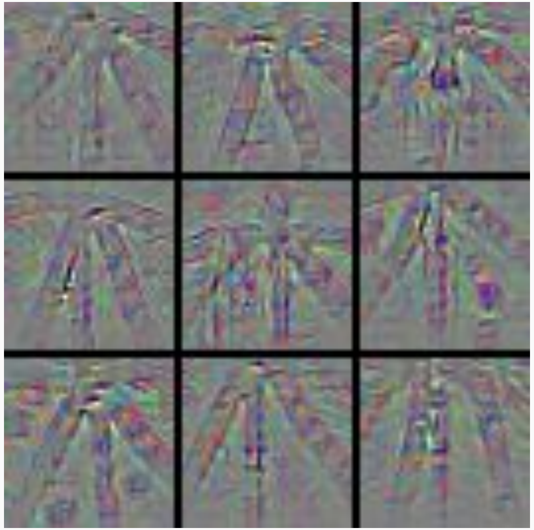
Deconvolution of the activated patch

- Input Patch for 73rd filter.
- Maximum activation : 893.476807
- Filter sensitive to edge transition.
- Classes with performance: train 78.57 crown 64.29 train 78.57 rabbit 78.57 crab 78.57 person sitting 71.43 rollerblades 92.86 pineapple 92.86 fish 92.86

inception_4a/1x1



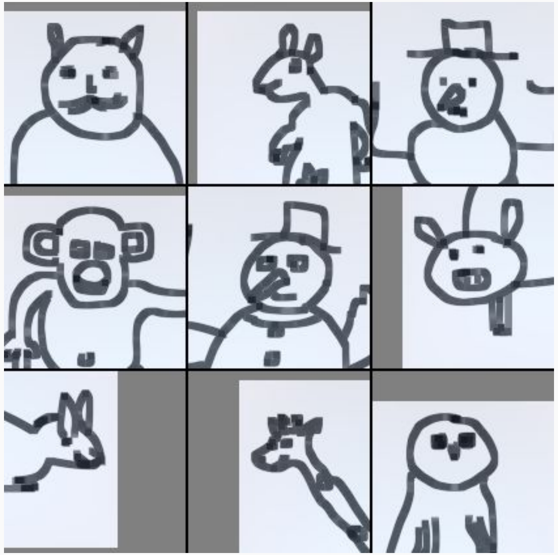
Input Patch for Maximum Activation



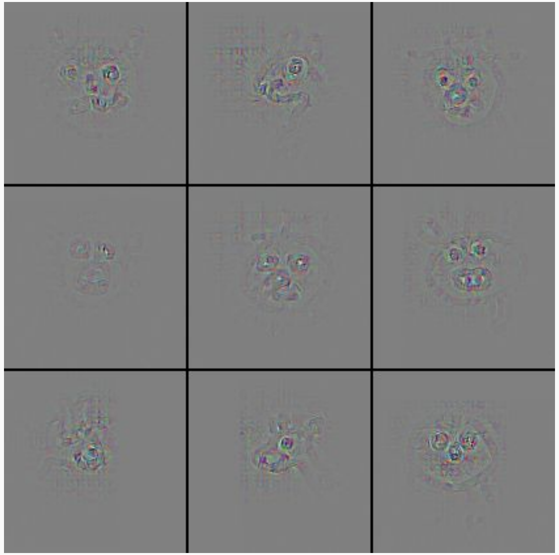
Deconvolution of the activated patch

- Input Patch for 79th filter
- Maximum activation : 800.211426
- Filter sensitive to triangular strokes.
- Classes with performance: umbrella 92.86 umbrella 92.86 umbrella 92.86 umbrella 92.86 umbrella 92.86 pizza 92.86 pizza 92.86 tent 92.86 sailboat 85.71

inception_4b/3x3



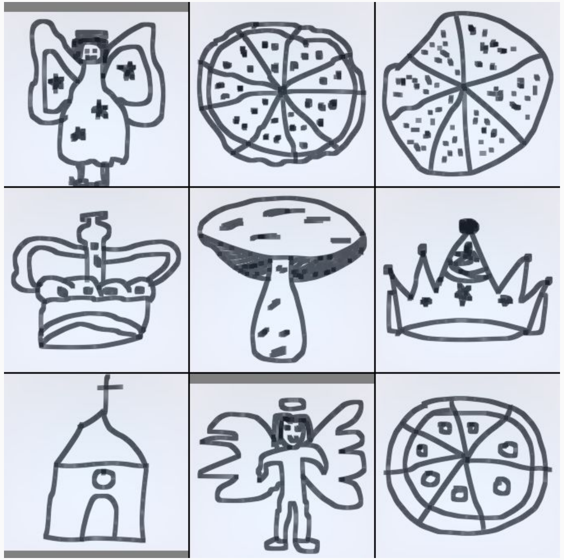
Input Patch for Maximum Activation



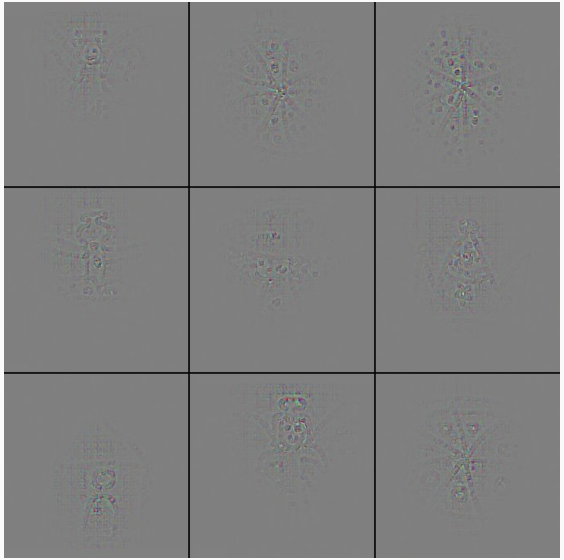
Deconvolution of the activated patch

- Input Patch for 87th filter
- Maximum activation : 515.481628
- Filter sensitive to animals.
- Classes with performance: cat 35.71 kangaroo 64.29 snowman 100.0 monkey 57.14 snowman 100.0 pig 64.29 squirrel 78.57 giraffe 100.0 owl 71.43

inception_4c/5x5



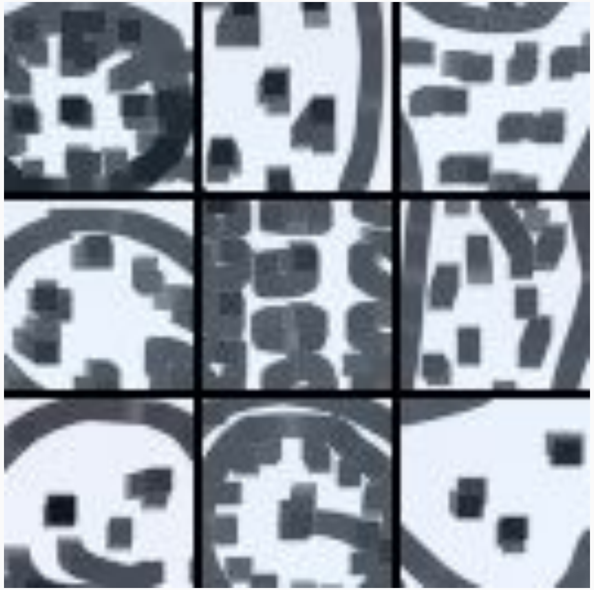
Input Patch for Maximum Activation



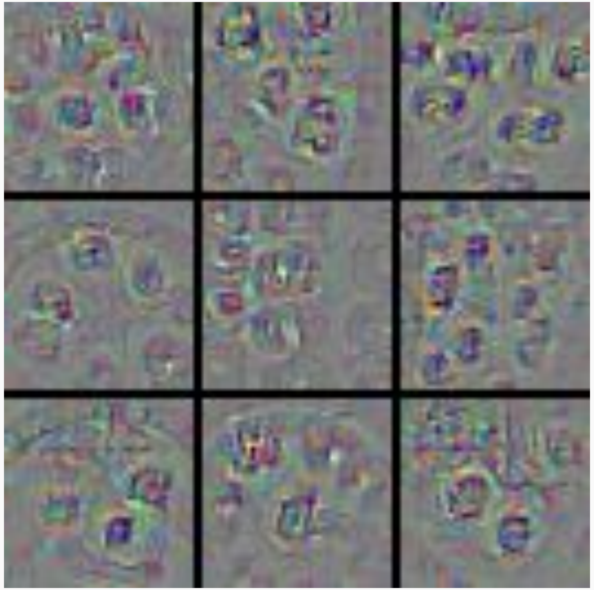
Deconvolution of the activated patch

- Input Patch for 45th filter
- Maximum activation : 611.922852
- Filter sensitive to some special classes.
- Classes with performance: angel 50.0 pizza 92.86 pizza 92.86 crown 64.29 mushroom 100.0 crown 64.29 church 100.0 angel 50.0 pizza 92.86

inception_4d/1x1



Input Patch for Maximum Activation

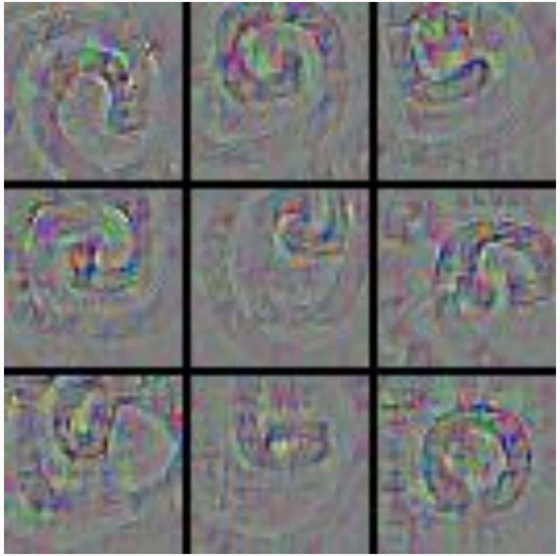


Deconvolution of the activated patch

- Input Patch for 97th filter
- Maximum activation : 375.478973
- Filter sensitive to dotted strokes.
- Classes with performance: wrist-watch 85.71 cactus 71.43 mermaid 71.43 telephone 71.43 telephone 71.43 cactus 71.43 person walking 100.0 wrist-watch 85.71 pizza 92.86

inception_4e/1x1

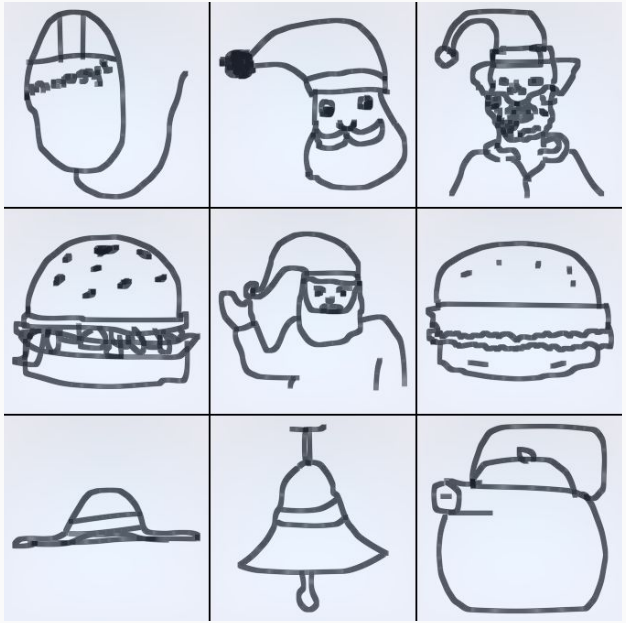
Input Patch for Maximum Activation



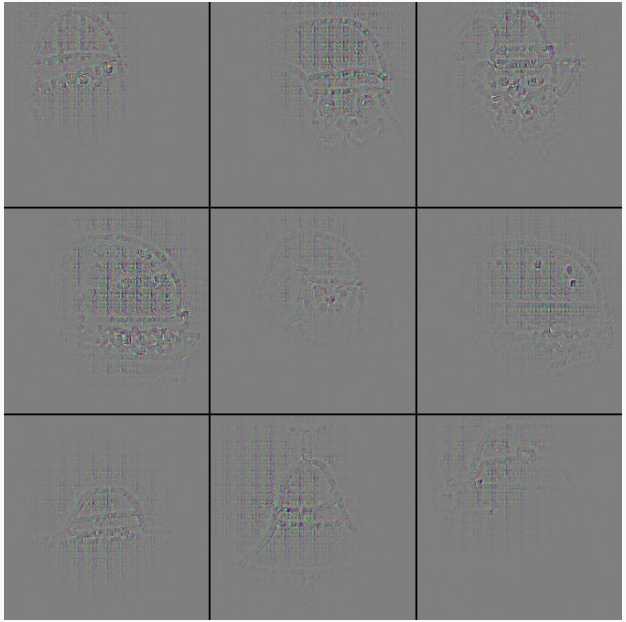
Deconvolution of the activated patch

- Input Patch for 12th filter
- Maximum activation : 291.59848
- Filter sensitive to spiral strokes.
- Classes with performance: snail 100.0 snail 100.0 snail 100.0 snail 100.0 snail 100.0 snail 100.0 snake 85.71 snail 100.0 camera 78.57

inception_5a/3x3



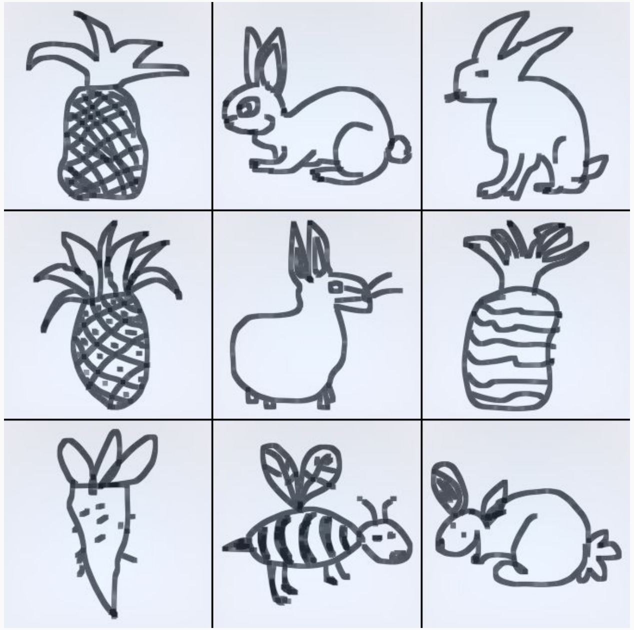
Input Patch for Maximum Activation



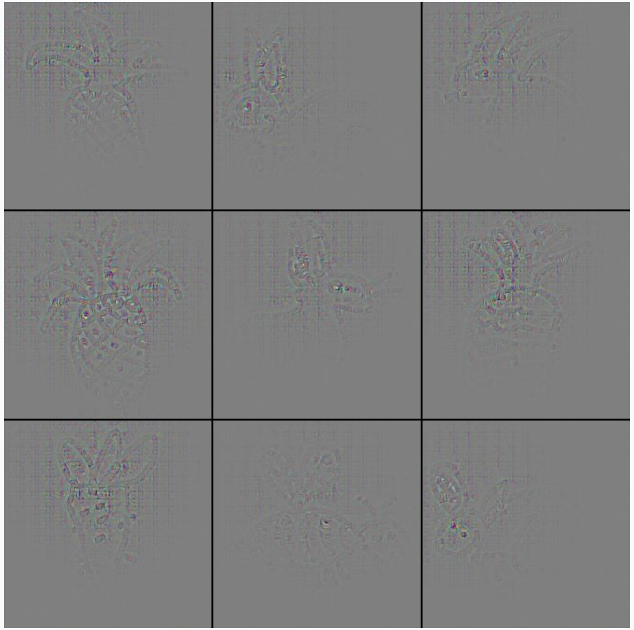
Deconvolution of the activated patch

- Input Patch for 228th filter
- Maximum activation : 163.456757
- Filter sensitive to specific classes.
- Classes with performance: computer-mouse 78.57 santa claus 78.57 santa claus 78.57 hamburger 85.71 santa claus 78.57 hamburger 85.71 hat 85.71 bell 85.71 teapot 100.0

inception_5b/5x5



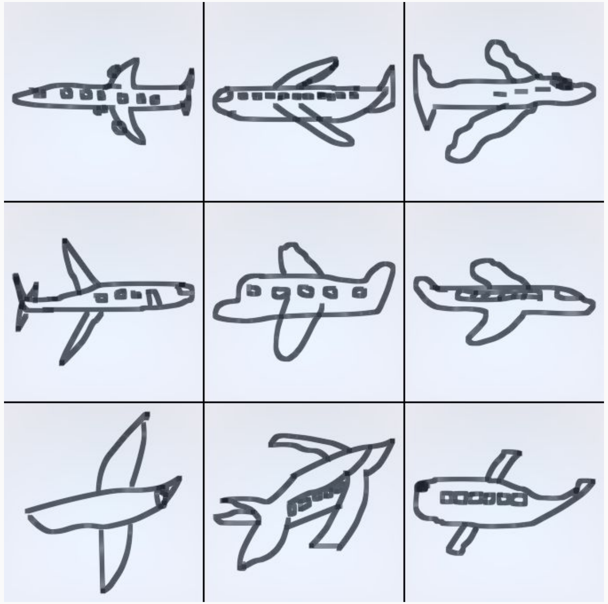
Input Patch for Maximum Activation



Deconvolution of the activated patch

- Input Patch for 89th filter
- Maximum activation : 71.704094
- Filter sensitive to some specific classes.
- Classes with performance: pineapple 92.86 rabbit 78.57 rabbit 78.57 pineapple 92.86 rabbit 78.57 pineapple 92.86 carrot 100.0 bee 57.14 rabbit 78.57

loss3/loss3



Input Patch for Maximum Activation

Deconvolution of the activated patch

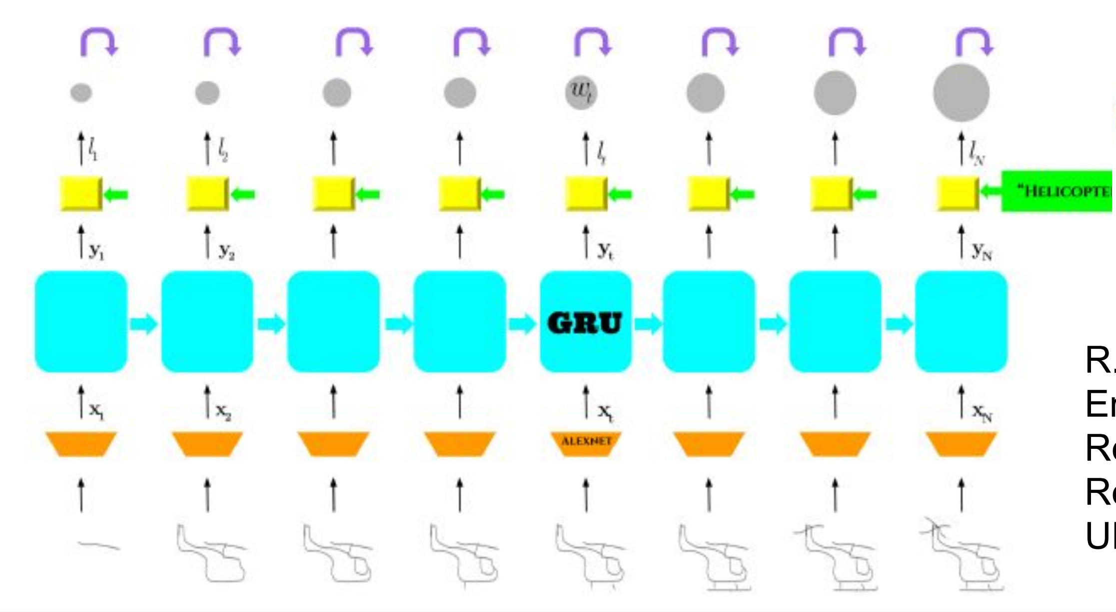
- Input Patch for 1st filter
- Maximum activation : 800.211426
- Filter corresponding to airplane class

Characterizing Visualizations in RNN

(AlexNet + GRU and AlexNet + LSTM architectures)

RNN for Sketch Recognition

- Sequential nature of stroke by stroke hand-sketching improves overall learning rate.
- GRU models sequential data in natural fashion.



$$r_t = \sigma(W_{xr}\mathbf{x_t} + W_{hr}h_{t-1} + b_r) \quad (1)$$

$$z_t = \sigma(W_{xz}\mathbf{x_t} + W_{hz}h_{t-1} + b_z) \quad (2)$$

$$\widetilde{h_t} = tanh(W_{xh}\mathbf{x_t} + U(r_t \odot h_{t-1}) + b_h) \quad (3)$$

$$h_t = (1 - z_t) \odot h_{t-1} + z_t \odot \widetilde{h_t} \quad (4)$$

$$\mathbf{y_t} = W_{hy}h_t \quad (5)$$

R.K. Sarvadevabhatla, J. Kundu & V. Babu R, Enabling My Robot To Play Pictionary: Recurrent Neural Networks For Sketch Recognition
URL: https://arxiv.org/pdf/1608.03369.pdf

Recognition Results for different networks

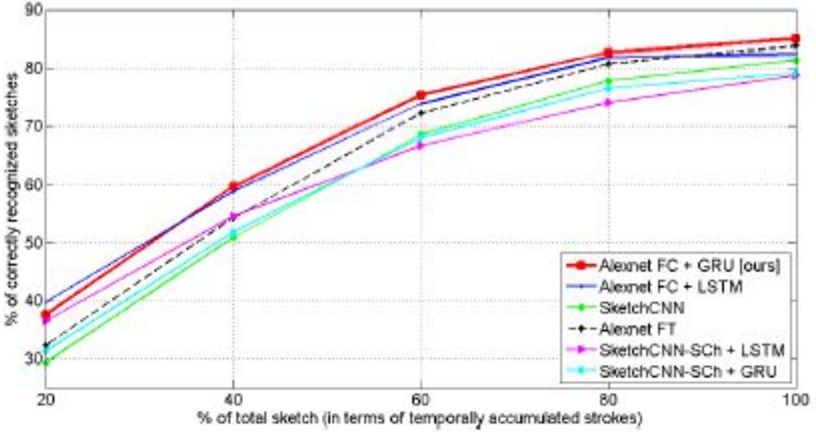


Figure 2: Comparison of online recognition performance for various classifiers. Our architecture recognizes the largest % of sketches at all levels of sketch completion. Best viewed in color.

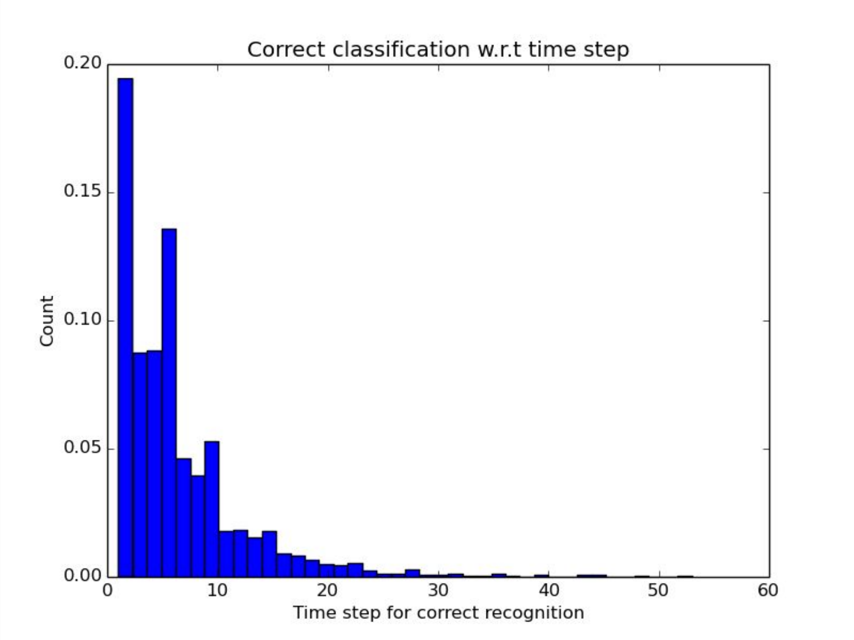
CNN	Recurrent Network	#Hidden	Avg. Acc
Alexnet-FC	**GRU**	**3600**	**85.1%**
Alexnet-FC	LSTM	3600	82.5%
SketchCNN [23]	-	-	81.4%
Alexnet-FT	-	-	83.9%
SketchCNN-Sch-FC	LSTM	3600	78.8%
SketchCNN-Sch-FC	GRU	3600	79.1%

Table 1: Average recognition accuracy (rightmost column) for various architectures. #Hidden refers to the number of hidden units used in recurrent network. We obtain state-of-the-art results for sketch object recognition.

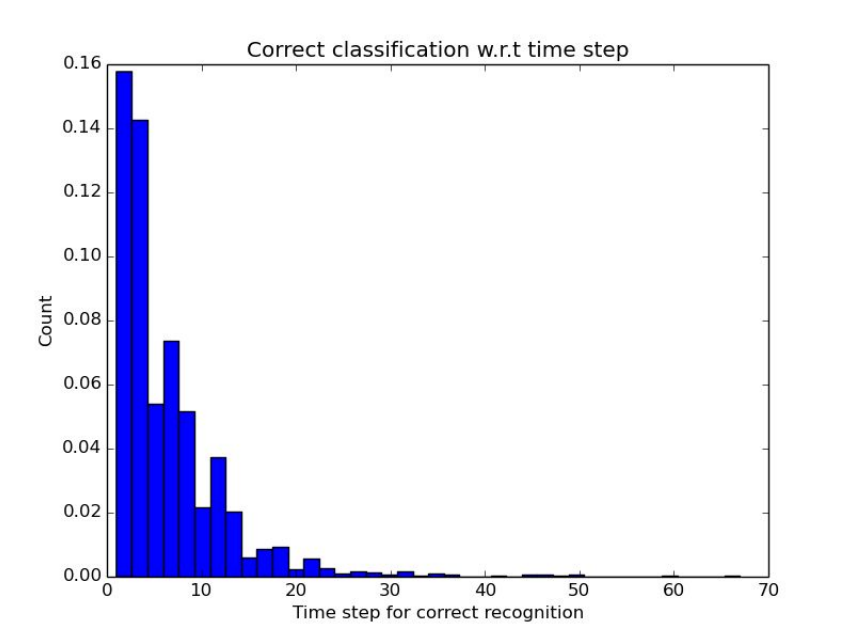
Accuracy

Accuracy	LSTM	GRU
Final prediction	80.13	82.72
Max pooling over predictions	77.90	79.42
Average pooling over predictions	75.71	77.19
Weighted Accuracy	80.58	83.39

Time Step Histogram



Alexnet + GRU

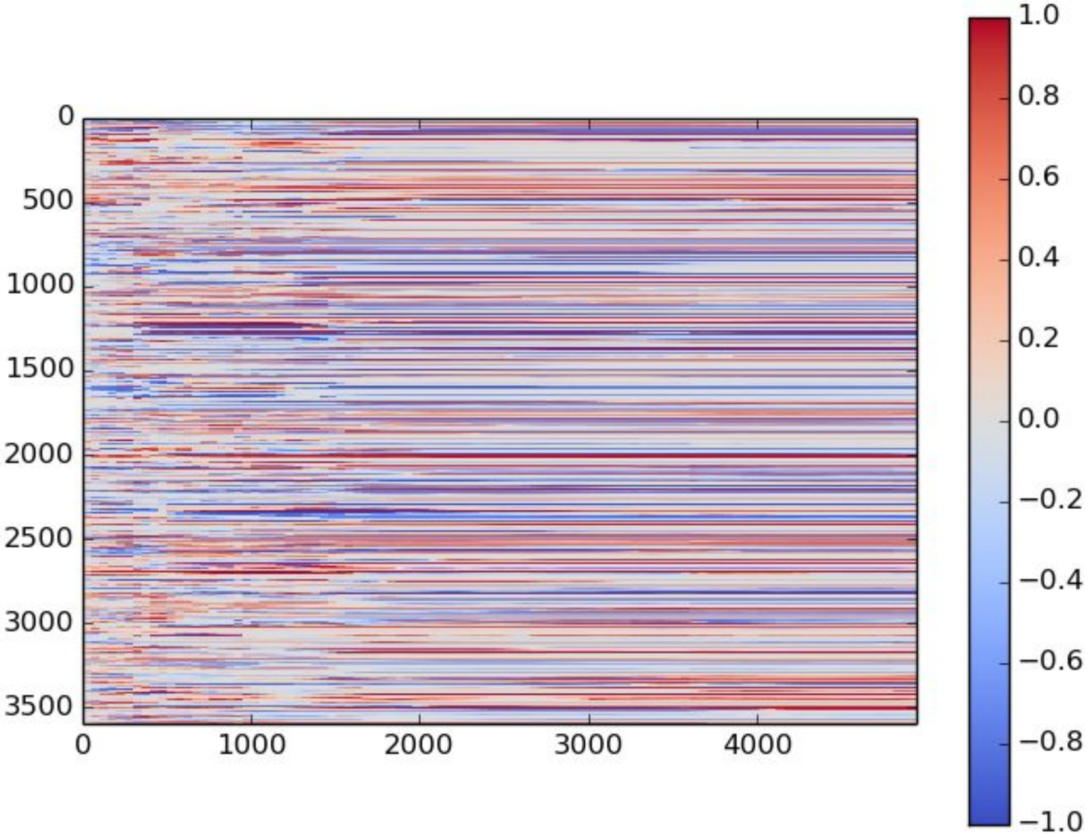


Alexnet + LSTM

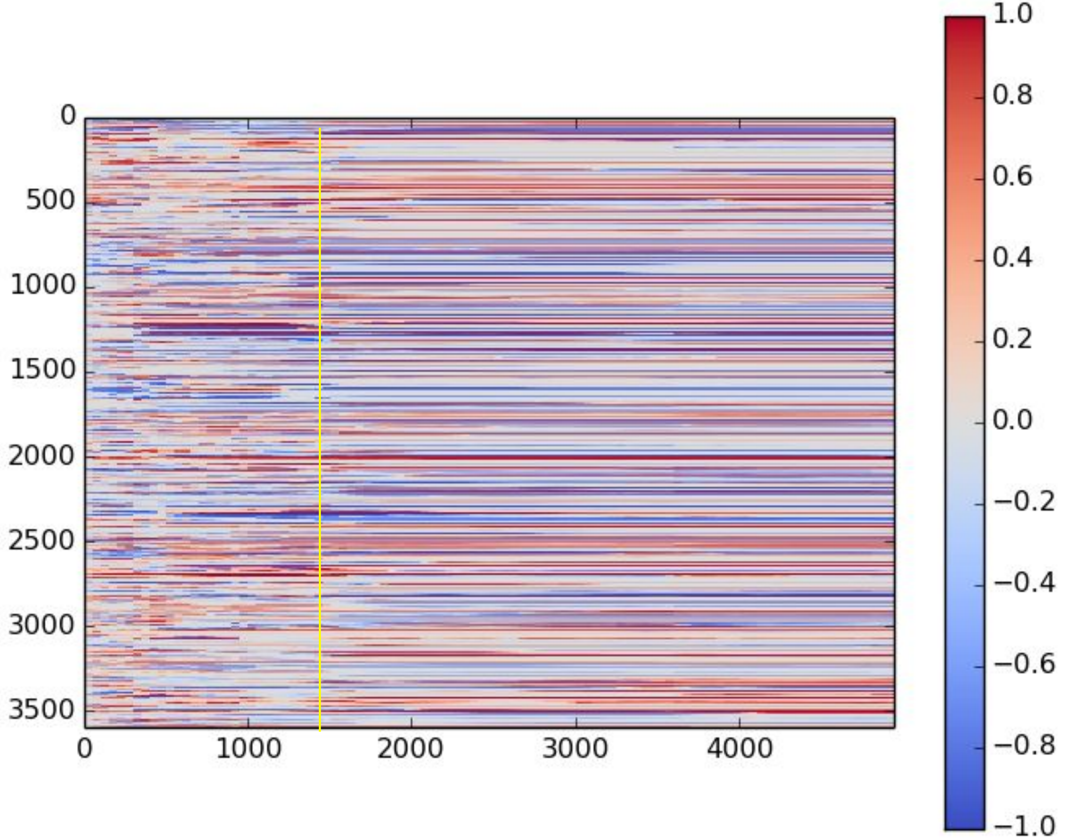
	Alexnet + GRU (80.13%)		Alexnet + LSTM (82.72%)	
	Categories	Accuracy(%)	Categories	Accuracy(%)
Top	backpack, baseball, candle, castle, giraffe, ladder, snail, spoon, t-shirt, tractor, trouser,zebra , etc.	100	Apple, bowl, candle, church, door, ear, envelope, giraffe, ladder, pear, sponge, t-shirt, wineglass	100
Mid	Book, crown, frog, mailbox, nose, rabbit, radio, saxophone, spider, trumpet,violin, windmill, etc.	71.43	Alarm, banana, cactus, flower, horse, knife, present, radio, scissors, train, wristwatch, etc.	78.57
Bottom	dog	28.57	computer-mouse	21.43

Predict at around which time sequence does the LSTM model predicts correct accuracy.

Class: Zebra

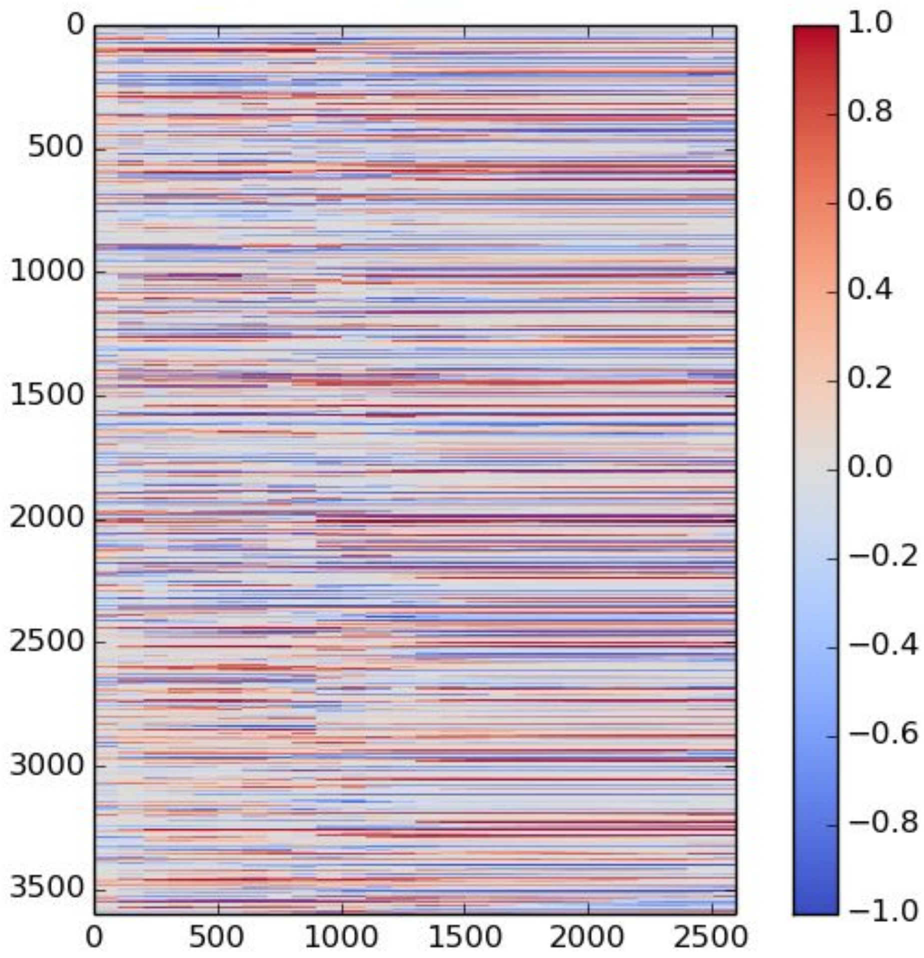


At t = 14 (xtick : 1400) , the model predicts it correct

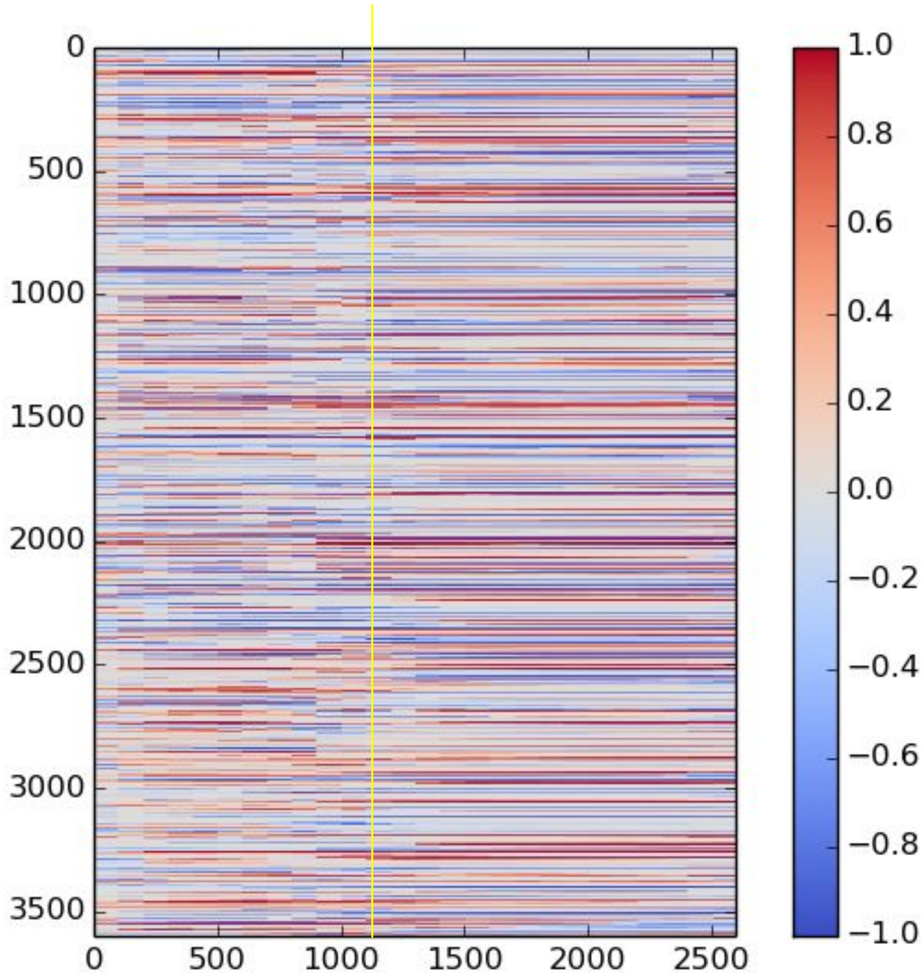


Predict at around which time sequence does the model predicts correct accuracy.

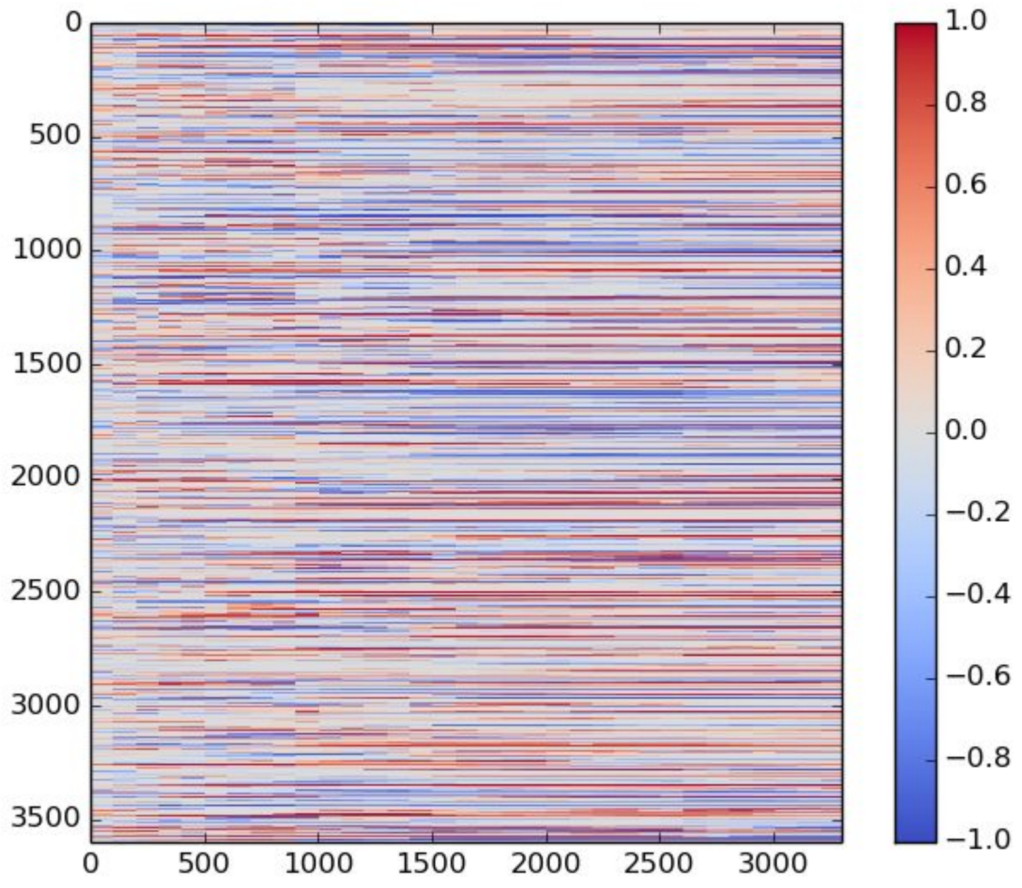
Class: crocodile



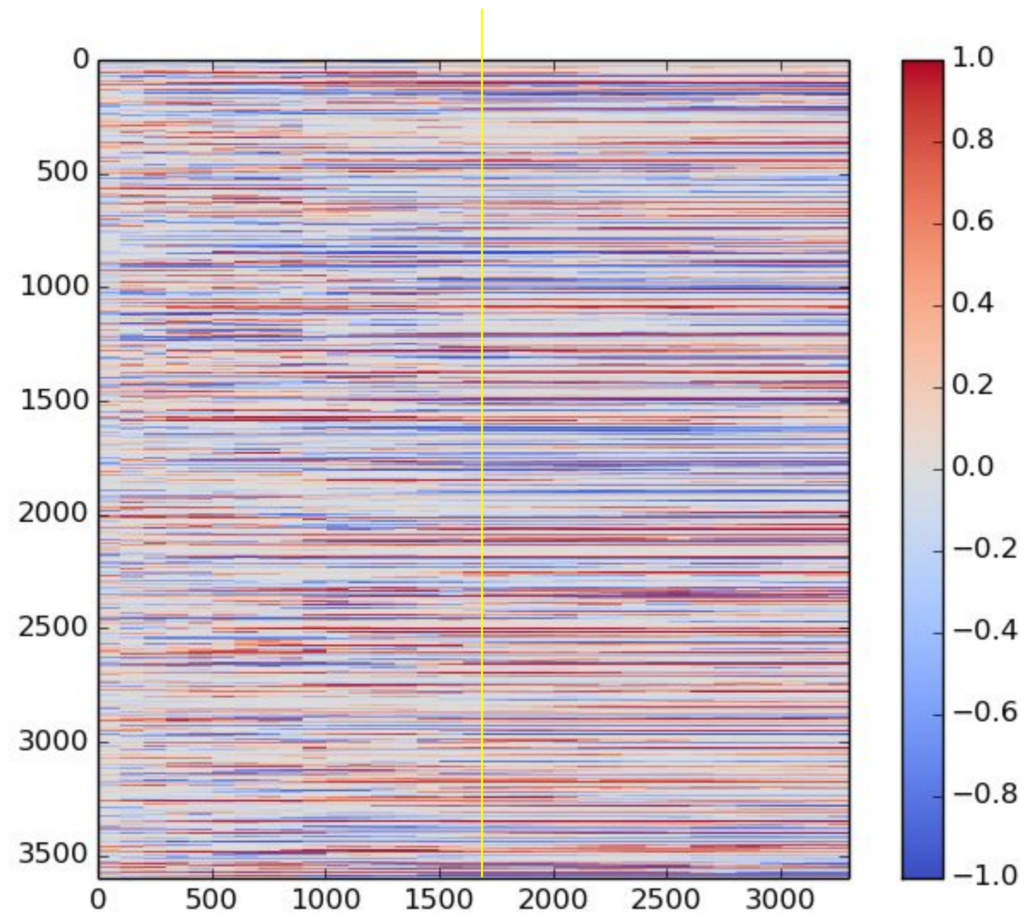
At t = 11 (xtick : 1100) , the model predicts it correct



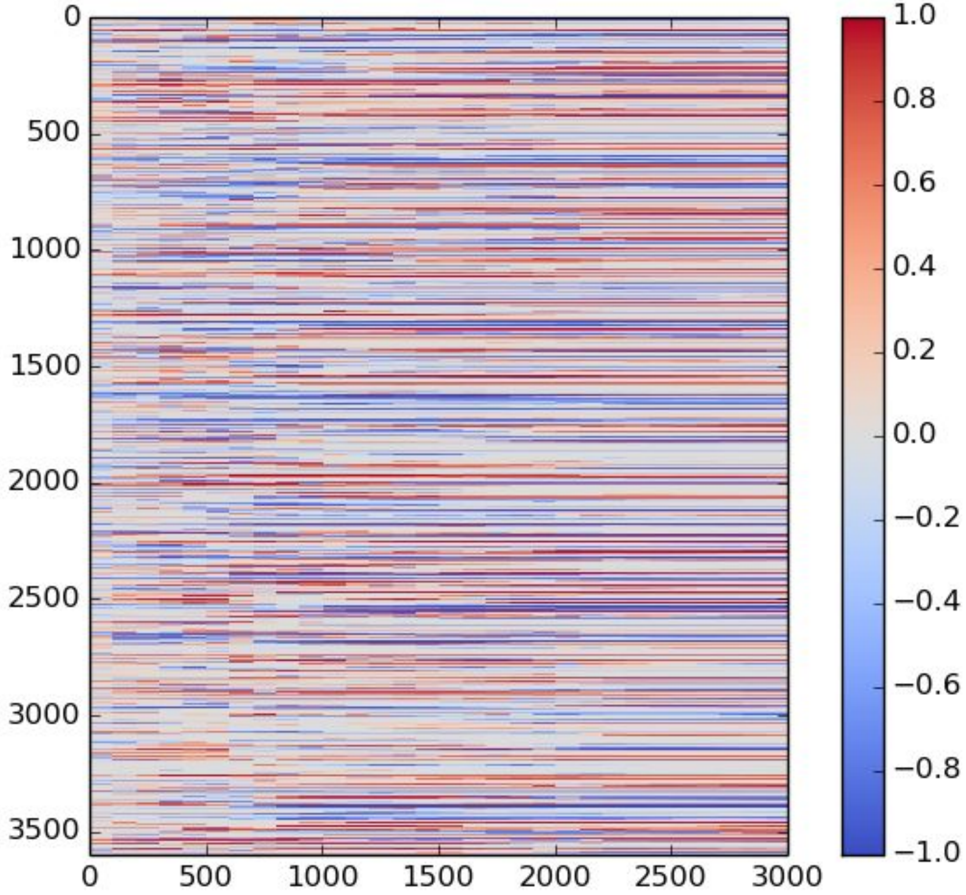
Predict at around which time sequence does the model predicts correct accuracy.



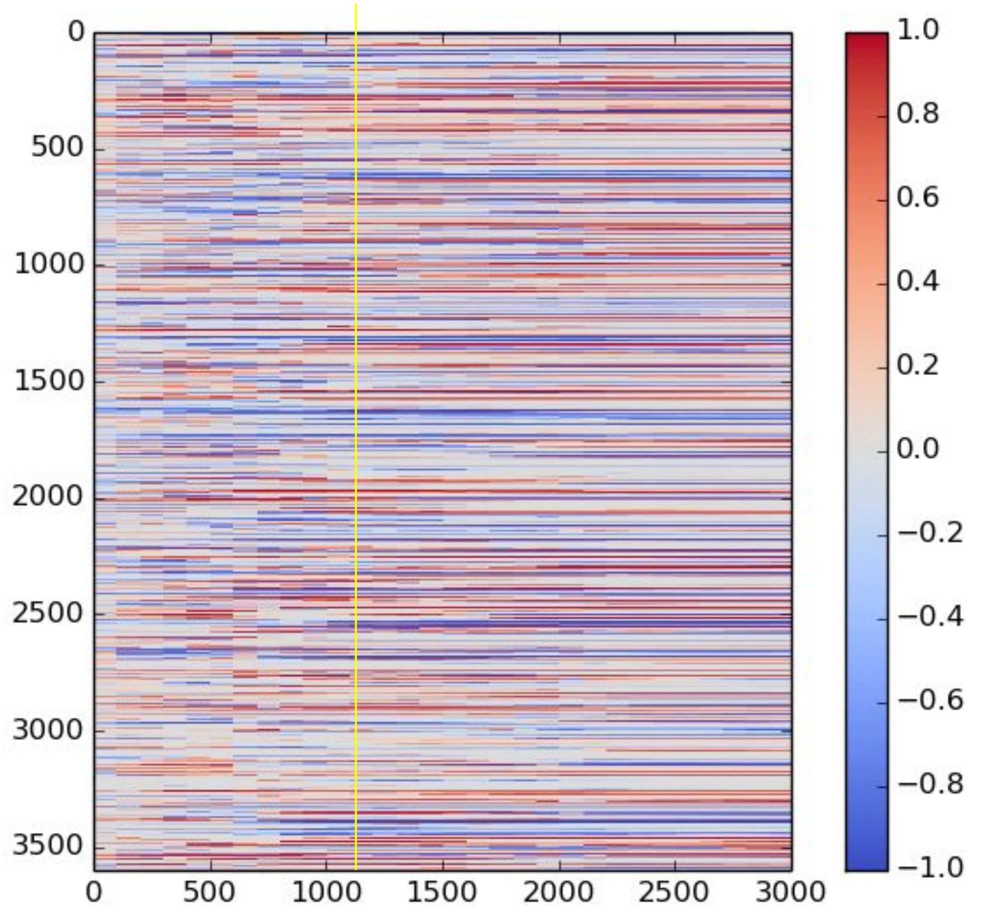
At t = 17 (xtick : 1700) , the model predicts it correct



Predict at around which time sequence does the model predicts correct accuracy.



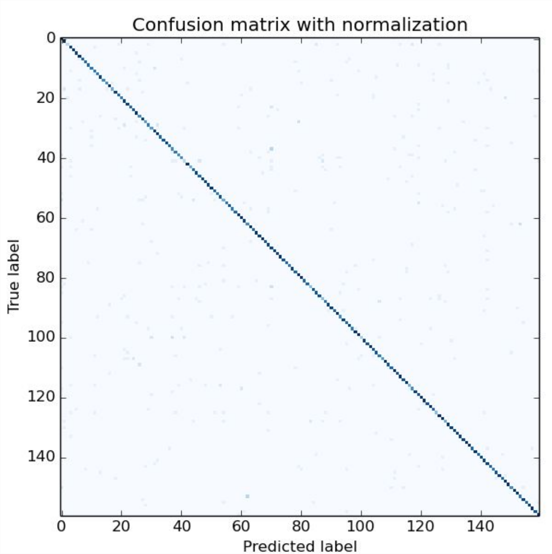
At t = 11 (xtick : 1100) , the model predicts it correct



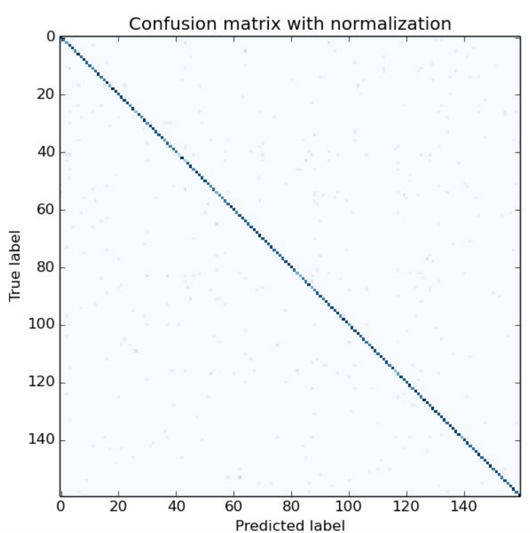
Comparing CNN with RNN

(CNN: AlexNet, RNN: LSTM, GRU)

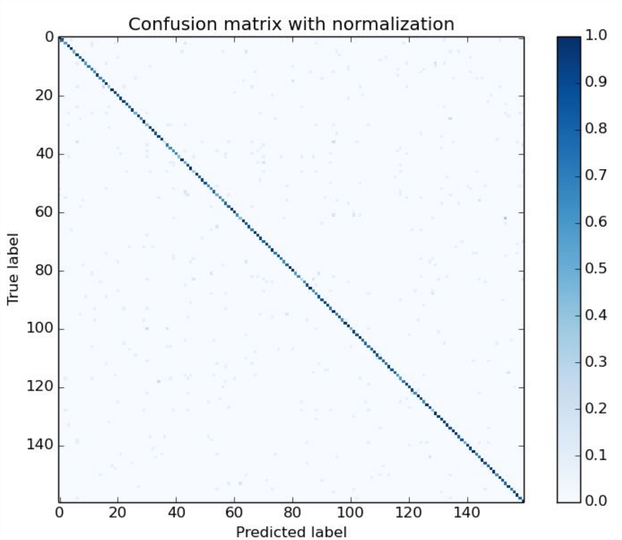
Comparing Confusion Matrices



AlexNet



AlexNet + GRU



AlexNet + LSTM

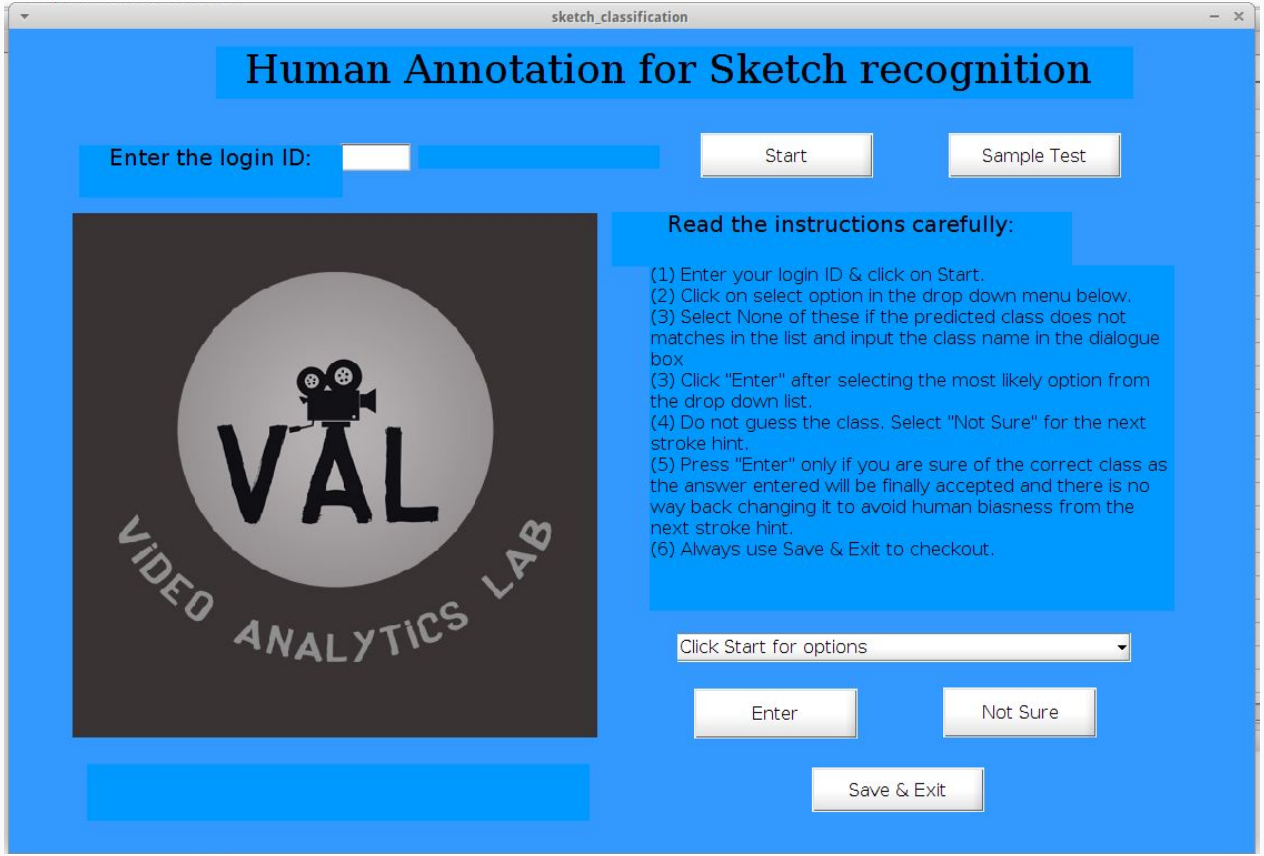
CNN vs RNN correctly classified sketches

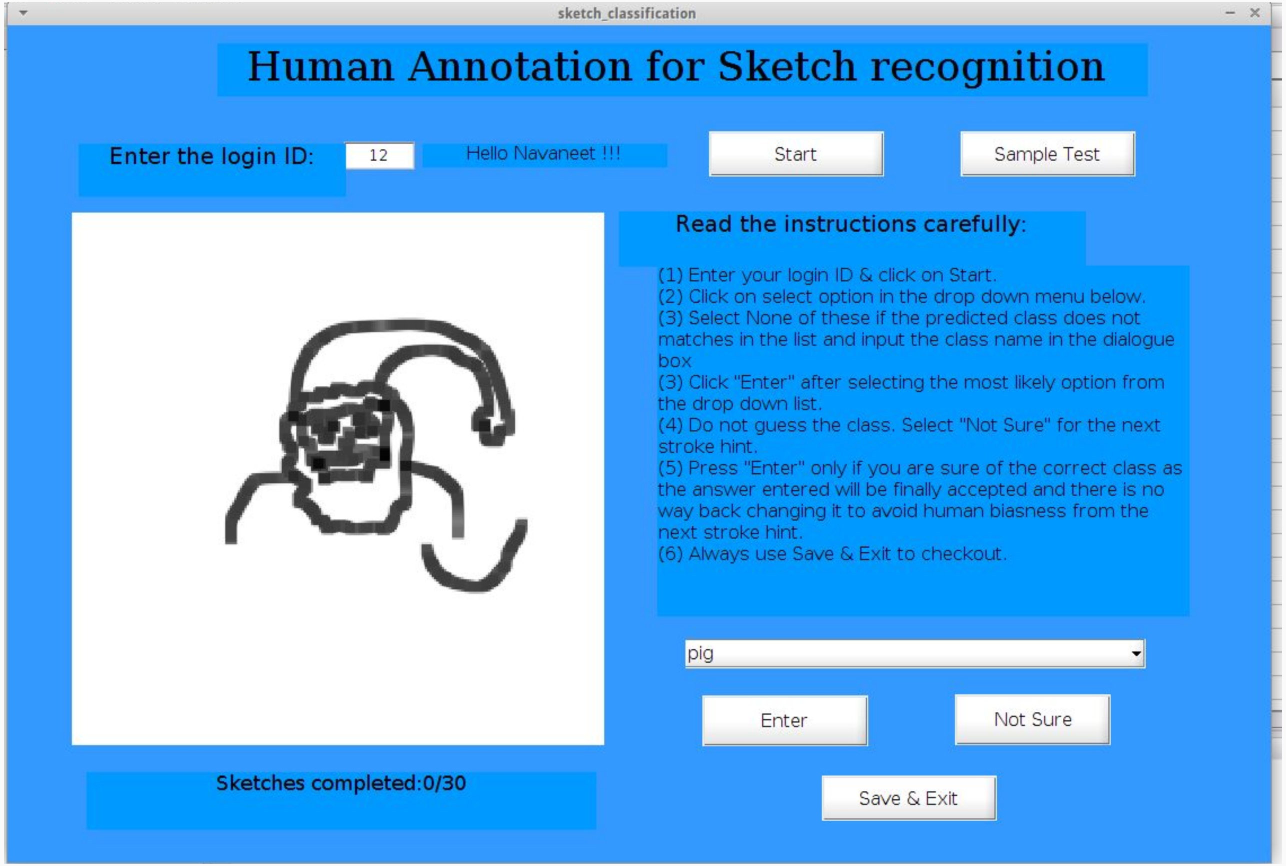
CNN: AlexNet	RNN: LSTM	No. of sketches
Correct	Correct	1732
Correct	Incorrect	173
Incorrect	Correct	63
Incorrect	Incorrect	272

CNN: AlexNet	RNN: GRU	No. of sketches
Correct	Correct	1772
Correct	Incorrect	133
Incorrect	Correct	81
Incorrect	Incorrect	254

Finding Human classification time steps from subjects through GUI

- Human evaluation setup:
  - Shortlist 10 random sketches from top-3 correct-from-first-stroke categories. Let the categories here be C-1
  - Shortlist 10 random sketches from top-3 correct-from-second-stroke categories. Let the categories here be C-2
  - Shortlist 10 random sketches from top-3 **misclassified** categories. Let the categories here be C-3.
  - Mix a random set of x other categories to have a total of 30 categories from the drop-down list. Include (x+1) as ‘Not Sure’ and make this default in the GUI tool.
  - GUI tool : Show subjects the sketches and ask them to make selection from this. Randomly shuffle the drop-down list each time.
  - Analysis:
    - Correctly recognized = More than half people correctly recognize a sketch.
      - How many C-1 category sketches are correctly recognized by people at first stroke?
      - How many C-2 category sketches are correctly recognized by people at second stroke?
      - How many C-3 category sketches are correctly recognized (at some stroke)?


sketch_classification
Human Annotation for Sketch recognition
Enter the login ID:
Start
Sample Test
Read the instructions carefully:
(1) Enter your login ID & click on Start.
(2) Click on select option in the drop down menu below.
(3) Select None of these if the predicted class does not matches in the list and input the class name in the dialogue box
(3) Click "Enter" after selecting the most likely option from the drop down list.
(4) Do not guess the class. Select "Not Sure" for the next stroke hint.
(5) Press "Enter" only if you are sure of the correct class as the answer entered will be finally accepted and there is no way back changing it to avoid human biasness from the next stroke hint.
(6) Always use Save & Exit to checkout.
VAL
VIDEO ANALYTICS LAB
Click Start for options
Enter
Not Sure
Save & Exit


sketch_classification
Human Annotation for Sketch recognition
Enter the login ID:
12
Hello Navaneet !!!
Start
Sample Test
Read the instructions carefully:
(1) Enter your login ID & click on Start.
(2) Click on select option in the drop down menu below.
(3) Select None of these if the predicted class does not matches in the list and input the class name in the dialogue box
(3) Click "Enter" after selecting the most likely option from the drop down list.
(4) Do not guess the class. Select "Not Sure" for the next stroke hint.
(5) Press "Enter" only if you are sure of the correct class as the answer entered will be finally accepted and there is no way back changing it to avoid human biasness from the next stroke hint.
(6) Always use Save & Exit to checkout.
pig
Enter
Not Sure
Sketches completed:0/30
Save & Exit

References

[1] Zeiler and Fergus paper

- https://www.cs.nyu.edu/~fergus/papers/zeilerECCV2014.pdf (paper)
- http://vision.cse.psu.edu/people/chrisF/deep-learning/DL_pres.pdf (slides)
- https://www.youtube.com/watch?v=ta5fdaqDT3M (video)

[2] A Taxonomy and Library for Visualizing Learned Features in CNNs (http://icmlviz.github.io/assets/papers/20.pdf)

[3]Yang et. al. , Sketch-a-Net; a Deep Neural network that beats humans : http://www.eecs.qmul.ac.uk/~yzs/yu2016sketchanet.pdf

[4] R.K. Sarvadevabhatla, J. Kundu & V. Babu R, Enabling My Robot To Play Pictionary: Recurrent Neural Networks For Sketch Recognition , URL: https://arxiv.org/pdf/1608.03369.pdf

[5] Yosinski's Deep Visualization Toolbox : https://github.com/yosinski/deep-visualization-toolbox

[6] LSTMViz toolbox http://blog.echen.me/2017/05/30/exploring-lstms/

[7] Visualizing and Understanding RNNs: https://arxiv.org/pdf/1506.02078

[8] Visualization Analysis for recurrent networks http://cslt.riit.tsinghua.edu.cn/mediawiki/images/6/6a/Visual.pdf

Thank You