Image classification of the illustration of everyday objects

By Jeffery Coulter, Aditya Joseph James, Manasi Swaminathan
Group 8

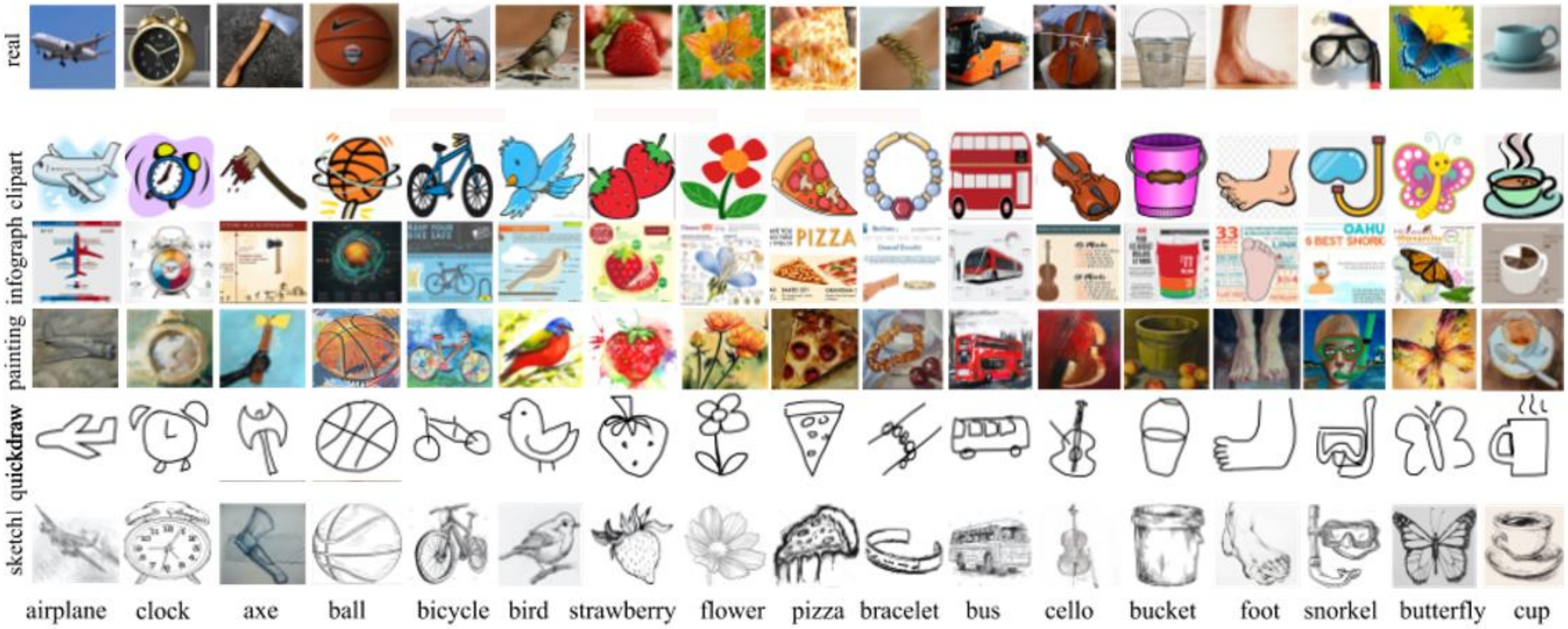
CSCI P556 Applied Machine Learning – Spring 2022
4th May'22

How well do image classification algorithms work on illustrations?

Classification using Domainnet data

Actual

Illustrations



345 Categories

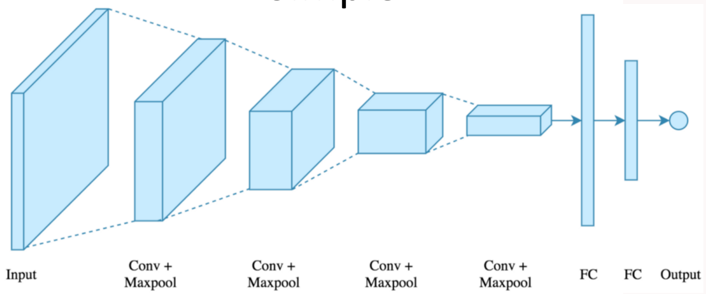
Image ref: http://ai.bu.edu/M3SDA/

Research Questions

1) Can a “simple” CNN architecture provide robust prediction power while classifying illustrations ?

2) How does the performance of a “simple” CNN architecture compare to the state-of-the-art architectures such as ResNet-50 and GoogLeNet ?

“Simple”

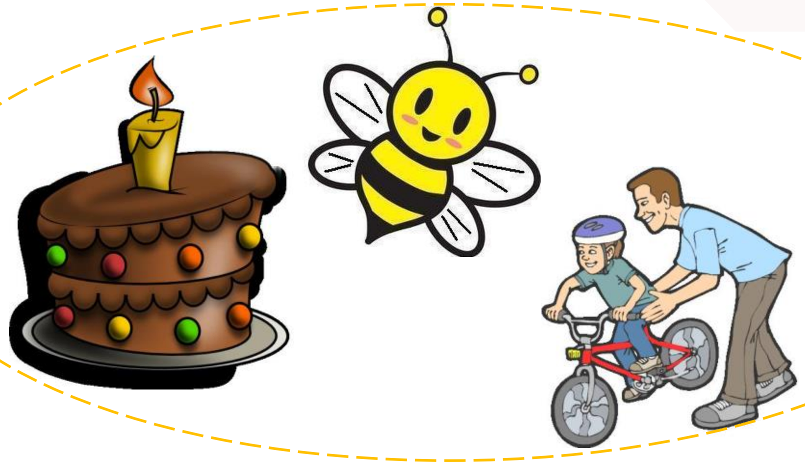


Vs

ResNet-50

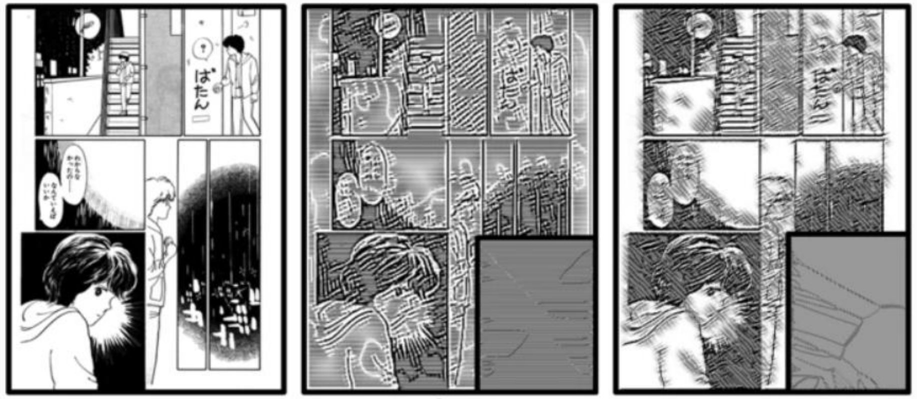
GoogLeNet

3) Can clustering algorithms be used to classify images based on color i.e. color image vs black and white?



https://miro.medium.com/max/1400/1*Bl-EM5EmdzGEdY34Ex_Ulw.png

Prior work



a b c

Image transformation results for two different neurons of the first convolutional layer. **a** Original image, **b** transformation via 7th neuron, **c** transformation via 11th neuron

[4]] Young-Min Kim, “Feature visualization in comic artist classification using deep neural networks.,” J. Big Data, vol. 6, pp. 56, 2019

[1] Xingchao Peng, Qinxun Bai, Xide Xia, Zijun Huang, Kate Saenko, and Bo Wang, “Moment matching for multi-source domain adaptation,” in Proceedings of the IEEE International Conference on Computer Vision, 2019, pp. 1406–1415.

[2] Yann LeCun and Corinna Cortes, “MNIST handwritten digit database,” 2010.

[3] Yanqing Zhou, Yongxu Jin, Anqi Luo, Szeyu Chan, Xiangyun Xiao, and Xubo Yang, “Toonnet: a cartoon image dataset and a dnn-based semantic classification system,” in Proceedings of the 16th ACM SIGGRAPH International Conference on Virtual-Reality Continuum and its Applications in Industry, VRCAI 2018, Hachioji, Japan, December 02-03, 2018, Koji Mikami, Zhigeng Pan, Matt Adcock, Daniel Thalmann, Xubo Yang, Tomoki Itamiya, and Enhua Wu, Eds. 2018, pp. 30:1–30:8, ACM.

[4] Young-Min Kim, “Feature visualization in comic artist classification using deep neural networks.,” J. Big Data, vol. 6, pp. 56, 2019.

[5] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott E. Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich, “Going deeper with convolutions,” CoRR, vol. abs/1409.4842, 2014.

[6] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, “Deep residual learning for image recognition,”CoRR, vol. abs/1512.03385, 2015.

[7] Gao Huang, Zhuang Liu, Laurens van der Maaten, and Kilian Q. Weinberger, “Densely connected convolutional networks,” 2018.

[8] Syed Mohammad Abid Hasan and Kwanghee Ko, “Depth edge detection by image-based smoothing and morphological operations,” Journal of Computational Design and Engineering, vol. 3, no. 3, pp. 191–197, 2016.

Classification of clipart: Architecture

------------------------ Model Summary ------------------------
Model: "Architecture 1"

Layer (type)	Output Shape	Param #
rescaling_1 (Rescaling)	(None, 64, 64, 1)	0
random_flip_1 (RandomFlip)	(None, 64, 64, 1)	0
conv2d_8 (Conv2D)	(None, 60, 60, 6)	156
max_pooling2d_5(MaxPooling 2D)	(None, 30, 30, 6)	0
dropout_8 (Dropout)	(None, 30, 30, 6)	0
conv2d_9 (Conv2D)	(None, 26, 26, 16)	2416
max_pooling2d_6(MaxPooling 2D)	(None, 13, 13, 16)	0
dropout_9 (Dropout)	(None, 13, 13, 16)	0
flatten_2 (Flatten)	(None, 2704)	0
dense_6 (Dense)	(None, 256)	692480
dropout_10 (Dropout)	(None, 256)	0
dense_7 (Dense)	(None, 128)	32896
dropout_11 (Dropout)	(None, 128)	0
dense_8 (Dense)	(None, 17)	2193

Total params: 730,141
Trainable params: 730,141
Non-trainable params: 0

CNN with 6,16 filters
Kernel size of 5x5
No Regularization
Relu activation

------------------------ Model Summary ------------------------
Model: "Architecure 2"

Layer (type)	Output Shape	Param #
rescaling_1 (Rescaling)	(None, 64, 64, 1)	0
random_flip_1 (RandomFlip)	(None, 64, 64, 1)	0
conv2d_10 (Conv2D)	(None, 62, 62, 16)	160
batch_normalization_6 (BatchNormalization)	(None, 62, 62, 16)	64
conv2d_11 (Conv2D)	(None, 60, 60, 16)	2320
batch_normalization_7 (BatchNormalization)	(None, 60, 60, 16)	64
max_pooling2d_7 (MaxPooling 2D)	(None, 30, 30, 16)	0
dropout_12 (Dropout)	(None, 30, 30, 16)	0
conv2d_12 (Conv2D)	(None, 28, 28, 32)	4640
batch_normalization_8 (BatchNormalization)	(None, 28, 28, 32)	128
conv2d_13 (Conv2D)	(None, 26, 26, 32)	9248
batch_normalization_9 (BatchNormalization)	(None, 26, 26, 32)	128
max_pooling2d_8 (MaxPooling 2D)	(None, 13, 13, 32)	0
dropout_13 (Dropout)	(None, 13, 13, 32)	0
conv2d_14 (Conv2D)	(None, 11, 11, 64)	18496
batch_normalization_10 (BatchNormalization)	(None, 11, 11, 64)	256
conv2d_15 (Conv2D)	(None, 9, 9, 64)	36928
batch_normalization_11 (BatchNormalization)	(None, 9, 9, 64)	256
max_pooling2d_9 (MaxPooling 2D)	(None, 4, 4, 64)	0
dropout_14 (Dropout)	(None, 4, 4, 64)	0
flatten_3 (Flatten)	(None, 1024)	0
dense_9 (Dense)	(None, 256)	262400
dropout_15 (Dropout)	(None, 256)	0
dense_10 (Dense)	(None, 128)	32896
dense_11 (Dense)	(None, 17)	2193

Total params: 370,177
Trainable params: 369,729
Non-trainable params: 448

CNN with 16,32, 64 filters
Kernel size of 3x3
Regularization
Relu activation

------------------------ Model Summary ------------------------
Model: "Architecure 3"

Layer (type)	Output Shape	Param #
rescaling_1 (Rescaling)	(None, 64, 64, 1)	0
random_flip_1 (RandomFlip)	(None, 64, 64, 1)	0
conv2d_16 (Conv2D)	(None, 63, 63, 16)	80
batch_normalization_12 (BatchNormalization)	(None, 63, 63, 16)	64
conv2d_17 (Conv2D)	(None, 62, 62, 16)	1040
batch_normalization_13 (BatchNormalization)	(None, 62, 62, 16)	64
max_pooling2d_10 (MaxPooling2D)	(None, 31, 31, 16)	0
dropout_16 (Dropout)	(None, 31, 31, 16)	0
conv2d_18 (Conv2D)	(None, 29, 29, 16)	2320
batch_normalization_14 (BatchNormalization)	(None, 29, 29, 16)	64
conv2d_19 (Conv2D)	(None, 27, 27, 16)	2320
batch_normalization_15 (BatchNormalization)	(None, 27, 27, 16)	64
max_pooling2d_11 (MaxPooling2D)	(None, 13, 13, 16)	0
dropout_17 (Dropout)	(None, 13, 13, 16)	0
conv2d_20 (Conv2D)	(None, 9, 9, 16)	6416
batch_normalization_16 (BatchNormalization)	(None, 9, 9, 16)	64
conv2d_21 (Conv2D)	(None, 5, 5, 16)	6416
batch_normalization_17 (BatchNormalization)	(None, 5, 5, 16)	64
max_pooling2d_12 (MaxPooling2D)	(None, 2, 2, 16)	0
dropout_18 (Dropout)	(None, 2, 2, 16)	0
flatten_4 (Flatten)	(None, 64)	0
dense_12 (Dense)	(None, 256)	16640
dropout_19 (Dropout)	(None, 256)	0
dense_13 (Dense)	(None, 128)	32896
dense_14 (Dense)	(None, 17)	2193

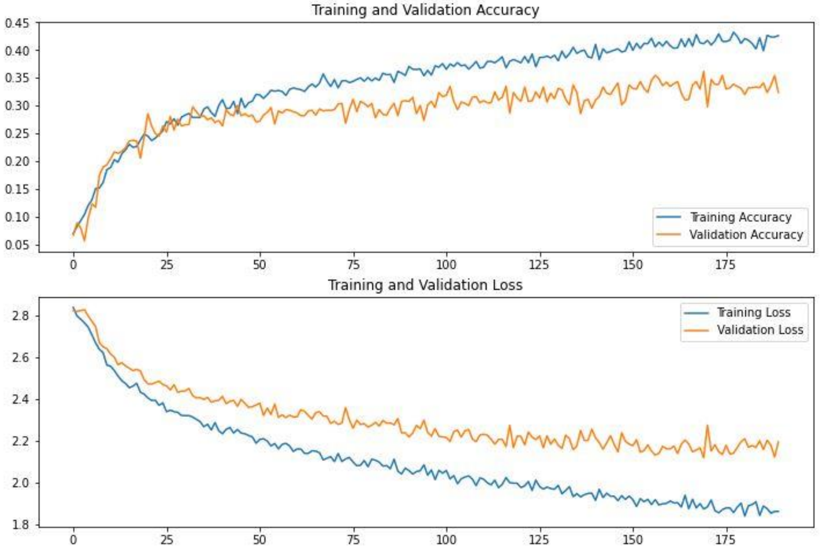
Total params: 70,705
Trainable params: 70,513
Non-trainable params: 192

CNN with 16 filters for each convolution
Kernel size of 2x2, 3x3, 5x5
Regularization
Relu activation

Classification of clipart: Loss and accuracy curves

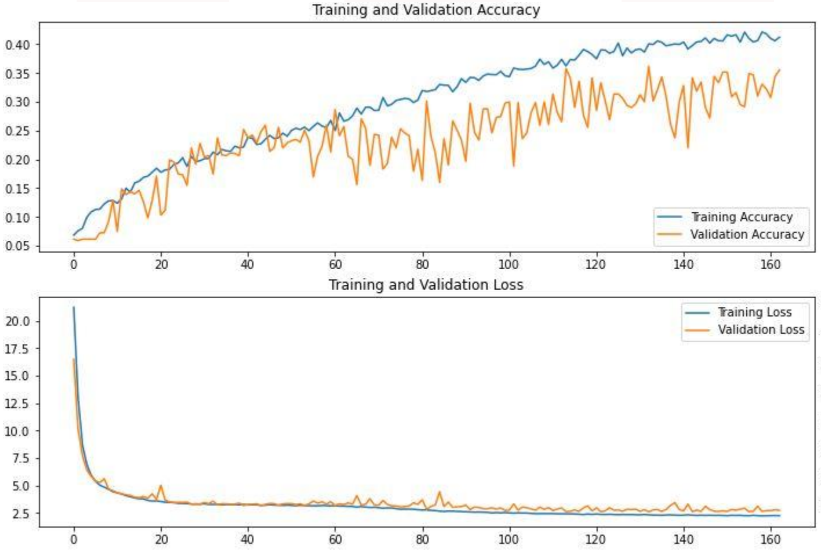
Model trained on 17 classes due to low sample volumes

Architecture: 1



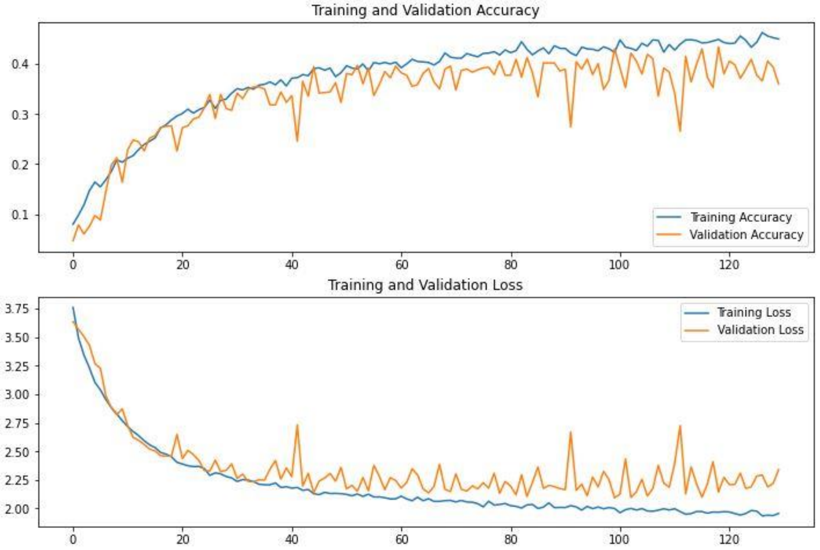
Epochs

Architecture: 2



Epochs

Architecture: 3



Epochs

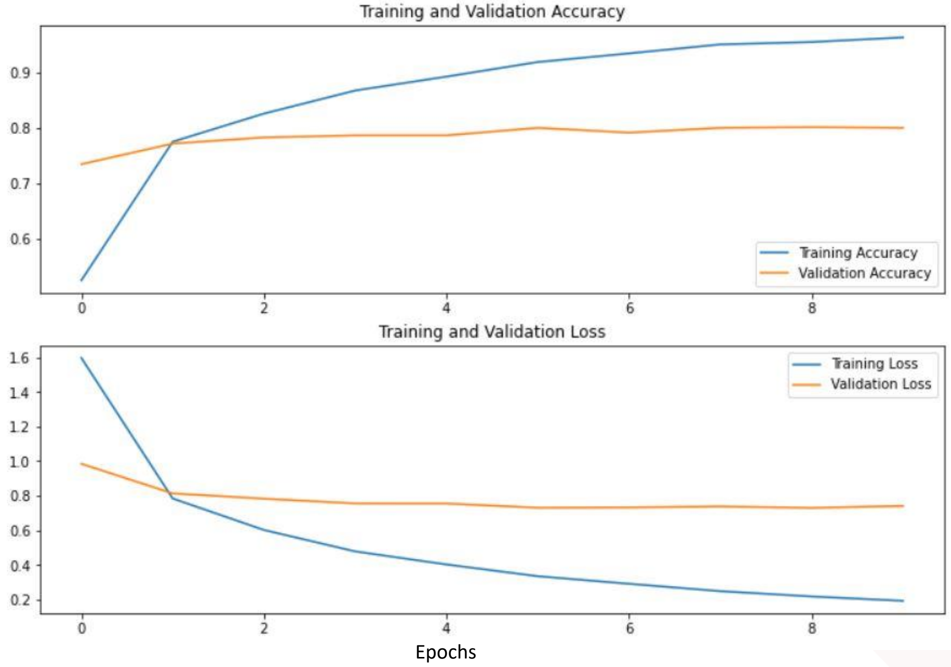
Training accuracy	42.6%	42.2%	45.02%
Validation accuracy	32.4%	35.5%	36.04%
Test accuracy	37.1%	35.8%	42.4%

Random guess of a class = 1/17 = 5.88%

Classification of clipart: ResNet-50

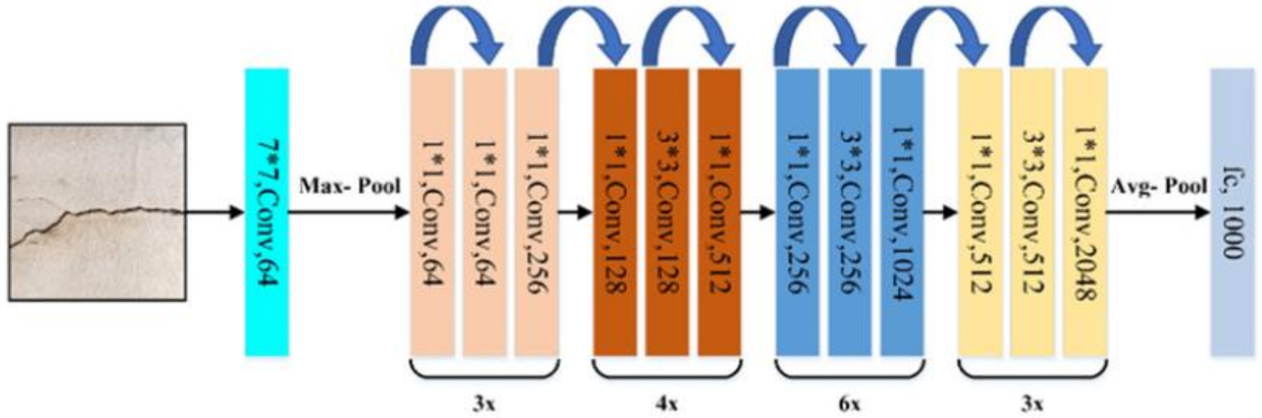
Transfer learning using weights from Imagenette

Transfer learning ResNet-50



Training accuracy	96.4%
Validation accuracy	80.02%
Test accuracy	78.5%

ResNet-50 Architecture

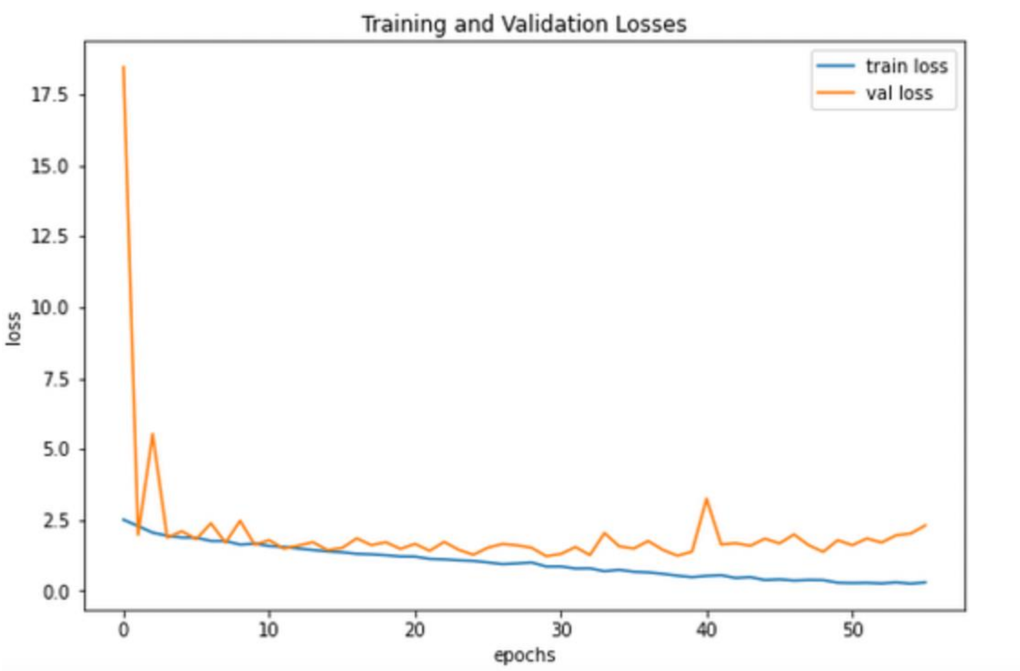


* The architecture of ResNet-50 model.

* Luqman Ali, Fady Alnajjar, Hamad Jassmi, Munkhjar-gal Gochoo, Wasif Khan, and Mohamed Serhani, “Performance evaluation of deep cnn-based crack detection and localization techniques for concrete structures,”Sensors, vol. 21, pp. 1688, 03 2021.

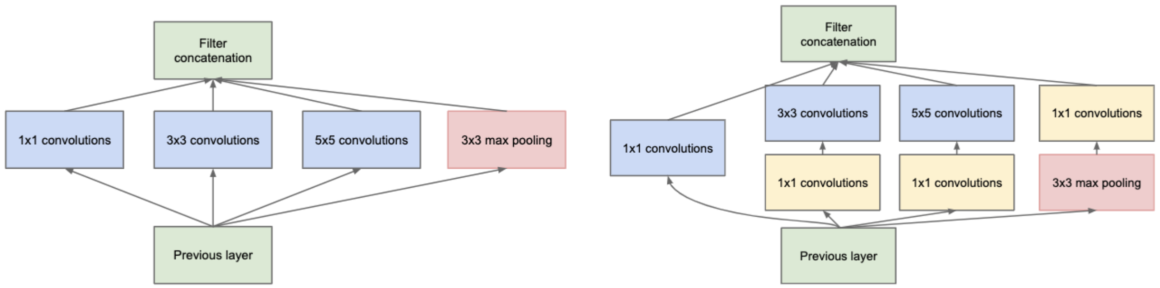
Classification of clipart: GoogLeNet

GoogLeNet



Training accuracy	91.8%
Validation accuracy	85.3%
Test accuracy	84.6%

type	patch size/ stride	output size	depth	#1×1	#3×3 reduce	#3×3	#5×5 reduce	#5×5	pool proj	params	ops
convolution	7×7/2	112×112×64	1							2.7K	34M
max pool	3×3/2	56×56×64	0								
convolution	3×3/1	56×56×192	2		64	192				112K	360M
max pool	3×3/2	28×28×192	0								
inception (3a)		28×28×256	2	64	96	128	16	32	32	159K	128M
inception (3b)		28×28×480	2	128	128	192	32	96	64	380K	304M
max pool	3×3/2	14×14×480	0								
inception (4a)		14×14×512	2	192	96	208	16	48	64	364K	73M
inception (4b)		14×14×512	2	160	112	224	24	64	64	437K	88M
inception (4c)		14×14×512	2	128	128	256	24	64	64	463K	100M
inception (4d)		14×14×528	2	112	144	288	32	64	64	580K	119M
inception (4e)		14×14×832	2	256	160	320	32	128	128	840K	170M
max pool	3×3/2	7×7×832	0								
inception (5a)		7×7×832	2	256	160	320	32	128	128	1072K	54M
inception (5b)		7×7×1024	2	384	192	384	48	128	128	1388K	71M
avg pool	7×7/1	1×1×1024	0								
dropout (40%)		1×1×1024	0								
linear		1×1×1000	1							1000K	1M
softmax		1×1×1000	0								



(a) Inception module, naïve version (b) Inception module with dimension reductions

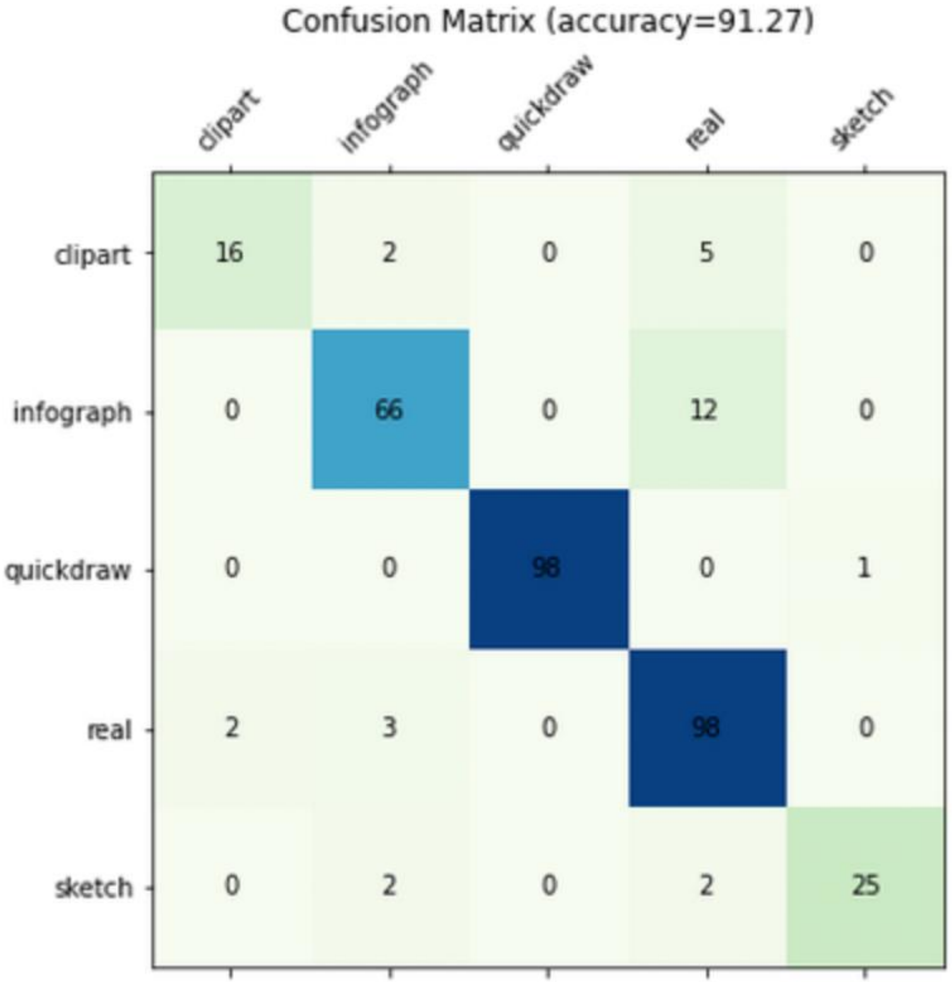
Figure 2: Inception module

ResNet-50 transfer learning and GoogLeNet are far superior architectures and could be used to solve illustration classification problems

Classification across categories using GoogLeNet

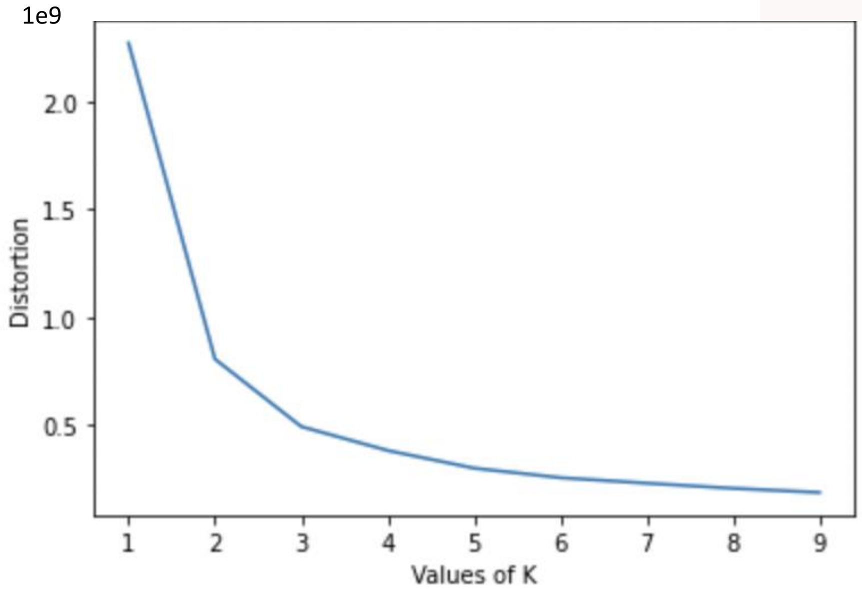
GoogLeNet

Training accuracy	99.3%
Validation accuracy	93.5%
Test accuracy	91.3%

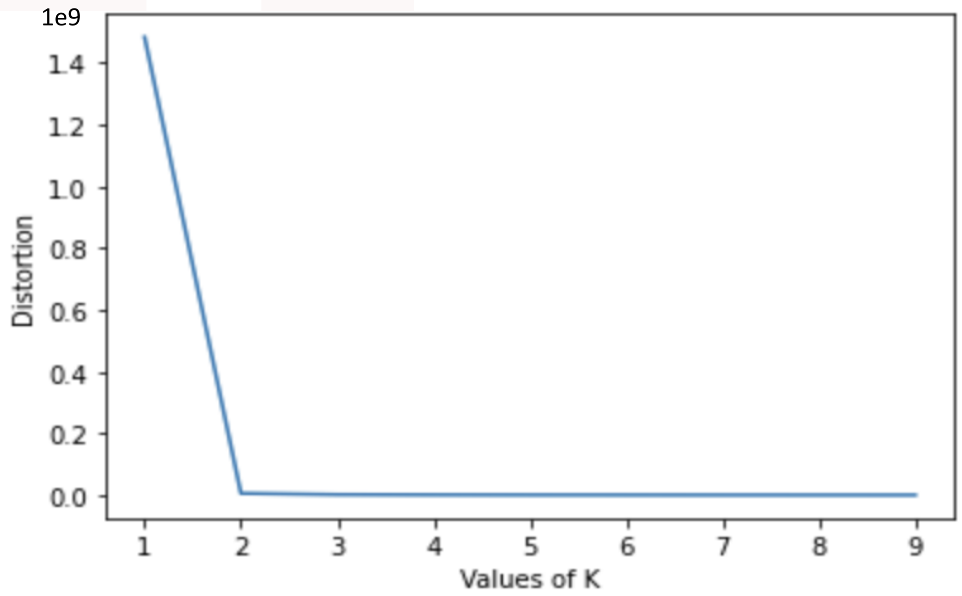


Clustering based on the dominant colors of an image: Approach

K Means elbow curve for colored image

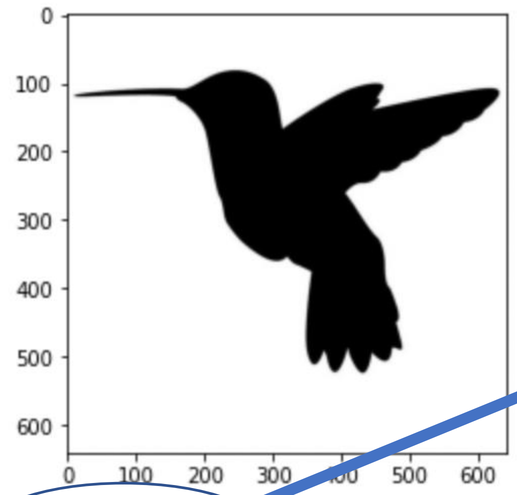


K Means elbow curve for black and white image



- The implementation is based on the hypothesis that the dominant colors in black and white image is black and white so the k value is set to 2. This is proved by the elbow method.

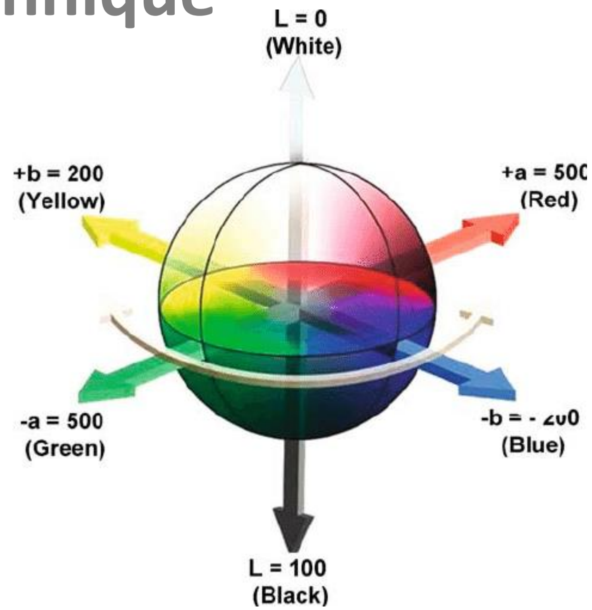
Clustering based on the dominant colors of an image: Technique




```
[[254 254 254]
 [  1   1   1]]
[0 0 0 ... 0 0 0]
#fefefe
#010101
black and white
```

Centroids/ RGB value of the 2 dominant colors

The RGB value is converted to CIE Labspace value



deltaE_cie76

`skimage.color.deltaE_cie76(lab1, lab2, channel_axis=-1)`

Euclidean distance between two points in Lab color space

Parameters

lab1 : array_like

reference color (Lab colorspace)

lab2 : array_like

comparison color (Lab colorspace)

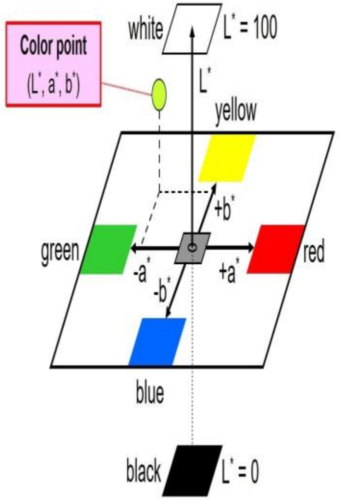
channel_axis : int, optional

This parameter indicates which axis of the arrays corresponds to channels.
New in version 0.19: `channel_axis` was added in 0.19.

Returns

dE : array_like

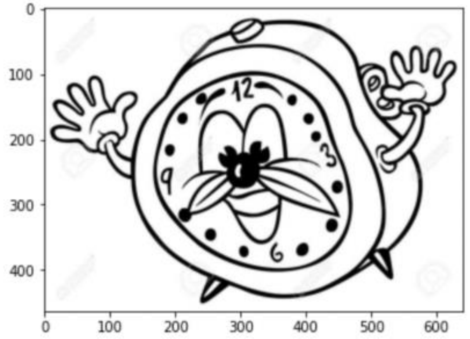
distance between colors *lab1* and *lab2*



https://scikit-image.org/docs/dev/api/skimage.color.html#skimage.color.deltaE_cie76
https://www.researchgate.net/figure/The-cubical-CIE-Lab-color-space_fig3_23789543
https://knowledge.ulprospector.com/10780/pc-the-cielab-lab-system-the-method-to-quantify-colors-of-coatings/

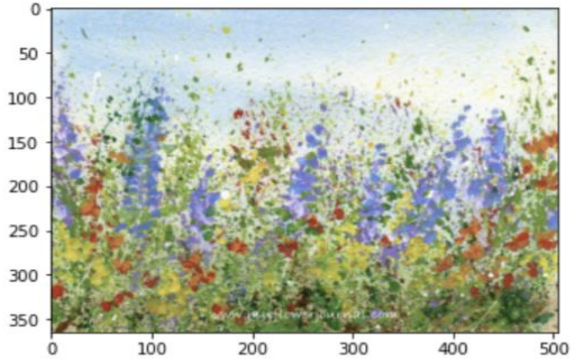
Clustering based on the dominant colors of an image: Results

Results for black and white image



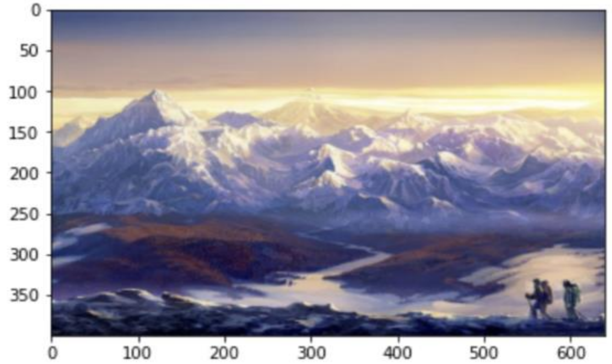

```
[[ 16  16  16]
 [251 251 251]]
[1 1 1 ... 1 1 1]
(463, 640, 3)
(463, 640, 3)
#101010
#fbfbfb
black and white
```

Results for color image




```
[[201 219 219]
 [128 137  78]]
[0 0 0 ... 0 0 0]
(365, 505, 3)
(365, 505, 3)
#c9dbdb
#80894e
color
```

Top 3 dominant colors (k=3)




```
[[119 121 158]
 [220 193 167]
 [ 51  52  95]]
[2 2 2 ... 2 2 2]
(400, 640, 3)
(400, 640, 3)
(400, 640, 3)
#77799e
#dcc1a7
#33345f
```

- K Means returns the centroid of the images. The centroids represent the RGB value. The RGB value is used to compare with the actual RGB value of black and white to determine if the image is black or white.

Conclusion

1. Through our experiments we find that transfer learning and GoogLeNet architecture are far superior to "simple" CNN architectures

2. We found that the same architectures perform with different accuracies when classifying between classes (clipart vs painting vs others) and within classes (aircraft vs ball vs bat, etc)

3. We were able to use the K Means algorithm to classify if an image was colored or black and white using the property that the centroids would represent the dominant colors of an image.

4. Additionally, the 3 dominant colors in a colored image is also visualized.

Thank You