Improving Object Characteristic Enumeration Models with Transfer Learning and Data Augmentation

Simon Swenson

4/20/2018

1 Abstract

Transfer learning is an important, growing area of deep learning research. Many ready-to-use networks exist on the internet and with pre-trained weights. One of our goals was to leverage that ever-growing set of ready-to-use networks, specifically image classification networks, and use them for generating sets of characteristics for an object from an image of that object.

We gathered a set of simple objects with known characteristics (like "blue" or "cube") and with some overlap in the characteristics (like a blue cube, a blue cylinder, and a red cube). To improve our model's understanding of the objects, we captured images of those objects in a variety of environments from a variety of angles.

To test if the models truly understood the objects and the corresponding characteristics, we introduced several test sets, (1) the training objects in a different environment, (2) objects which shared some characteristics of the training objects in the same environment(s), and (3) objects which shared some characteristics of the training objects in a different environment.

Our findings were that many model configurations descended to the all 0's output model. We observed that, as the number of neurons per layer and number of densely connected layers in our network increased, the probability of descending to the all-0's output model approached 1. However, some model configurations were able to improve accuracy on the training data, but no model was able to find any connection between the training data and testing data; as the training accuracy increased, test accuracy was immediately inversely affected. Possible reasons are given in the "Conclusion" section.

2 Introduction

Though most interest in image processing applications has been directed toward one of:

- Object detection
- Object classification
- Image captioning

an important task for understanding objects as they are is to understand the characteristics of that object and be able to abstract those characteristics and apply them to other, somewhat related, objects. For example, after seeing enough instances of objects which have a characteristic of "blue," one might come to an understanding of that characteristic and would thus be able to apply the characteristic "blue" to other blue objects that one has never experienced prior. This ability is essential for robots and AI agents, as it allows them to apply their understanding of previous objects to new objects.

Such a task can be quite difficult, and a large, complicated neural network may be required for it. However, many ready-to-use models with pre-computed weights exist in libraries such as Keras. As the task outlined above requires image processing, the relevant ready-to-use models must also take image data as an input. Image classification networks, such as "Xception," are promising in that regard, and we opted to make use of them.

In addition to such transfer learning, we can turn to the human learning process to inform us on how to proceed. One method that humans often use to learn about new objects is to examine them from a variety of different angles. Thus, such an approach seemed promising for this task.

3 Architecture

Let:

- $O = \{o_1, o_2, ..., o_n\}$ be the set of objects to be considered
- $O_{tr} \subseteq O$ be the set of train objects
- $O_{te} \subseteq O$ be the set of test objects with $O_{tr} \cap O_{te} = \emptyset$

- c be a set, the corpus of characteristics found among training objects for this session
- $C : O \to \wp(c)$ be a mapping from objects to their associated characteristics.
- $L = \{l_1, l_2, ..., l_m\}$ be the set of locations in the world at which to capture objects
- $L_{tr} \subseteq L$ be the set of train locations
- $L_{te} \subseteq L$ be the set of test locations with $L_{tr} \cap L_{te} = \emptyset$
- P be a set of two-tuples, a rotation and a translation from which the objects will be captured at each location

Then we can define the image capture, I, as a mapping $I : O \times L \times P \to R^{q \times r}$, for some q, r. Our goal is then to find the mapping $M : R^{q \times r} \to \wp(c)$ such that $\forall o \in O, l \in L, p \in P, M(I(o, l, p)) = C(o)$. However, using transfer learning requires us to not only find one mapping M, but two models $Mo_1, Mo_2 | M = Mo_2 \circ Mo_1$.

3.1 Test Criteria

An important part of this task is defining what, exactly, to test, as it will determine the exact architecture for the solution. For this task, we decided to measure the success of the model in three different ways:

1. Given an image of an object:
   (a) in the set of training objects but
   (b) in a different environment from the training environments

   can the network produce the correct characteristics for that object?
2. Given an image of an object:
   (a) not in the set of training objects but
   (b) in an environment in the set of training environments and
   (c) sharing at least one characteristic with an object in the set of training objects

   can the network produce the correct characteristics for that object?
3. Given an image of an object
   (a) not in the set of training objects and
   (b) in an environment in the set of training environments and
   (c) sharing at least one characteristic with an object in the set of training objects

   can the network produce the correct characteristics for that object?

In formal terms, we measure success by computing the accuracy of our model, Mo_2 using the following three metrics:

1. $Accur_{Te_1} = \frac{\sum_{o \in O_{tr}} \sum_{l \in L_{te}} \sum_{p \in P} |c| - |Mo_2(Mo_1(I(o,l,p))) \oplus C(o)|}{|O_{tr}||L_{te}||P||c|}$
2. $Accur_{Te_2} = \frac{\sum_{o \in O_{te}} \sum_{l \in L_{tr}} \sum_{p \in P} |c| - |Mo_2(Mo_1(I(o,l,p))) \oplus C(o)|}{|O_{te}||L_{tr}||P||c|}$
3. $Accur_{Te_3} = \frac{\sum_{o \in O_{te}} \sum_{l \in L_{te}} \sum_{p \in P} |c| - |Mo_2(Mo_1(I(o,l,p))) \oplus C(o)|}{|O_{te}||L_{te}||P||c|}$

4 Implementation

4.1 Data Acquisition

We used the following objects for training and testing.:

Obj	Shape	Dimensions	Color	Characteristics	Te/Tr
o_{bo}	Cone	d = 82.5 mm h = 82.5 mm	Krylon Gloss True Blue	{"blue", "cone"}	Tr
o_{bu}	Cube	w = 60 mm	Krylon Gloss True Blue	{"blue", "cube"}	Tr
o_{by}	Cylinder	d = 75 mm h = 50 mm	Krylon Gloss True Blue	{"blue", "cylinder"}	Tr
o_{ro}	Cone	d = 82.5 mm h = 82.5 mm	Krylon Gloss Banner Red	{"red", "cone"}	Tr
o_{ru}	Cube	w = 60 mm	Krylon Gloss Banner Red	{"red", "cube"}	Tr
o_{ry}	Cylinder	d = 75 mm h = 50 mm	Krylon Gloss Banner Red	{"red", "cylinder"}	Tr
o_{yo}	Cone	d = 82.5 mm h = 82.5 mm	Krylon Gloss Sun Yellow	{"yellow", "cone"}	Tr
o_{yu}	Cube	w = 60 mm	Krylon Gloss Sun Yellow	{"yellow", "cube"}	Tr
o_{yy}	Cylinder	d = 75 mm h = 50 mm	Krylon Gloss Sun Yellow	{"yellow", "cylinder"}	Tr
o_{yi}	Icosahedron	r_{flats} = 70 mm	Krylon Gloss Sun Yellow	{"yellow"}	Te
o_{wu}	Cube	w = 80 mm	Rust-oleum Flat White	{"cube"}	Te

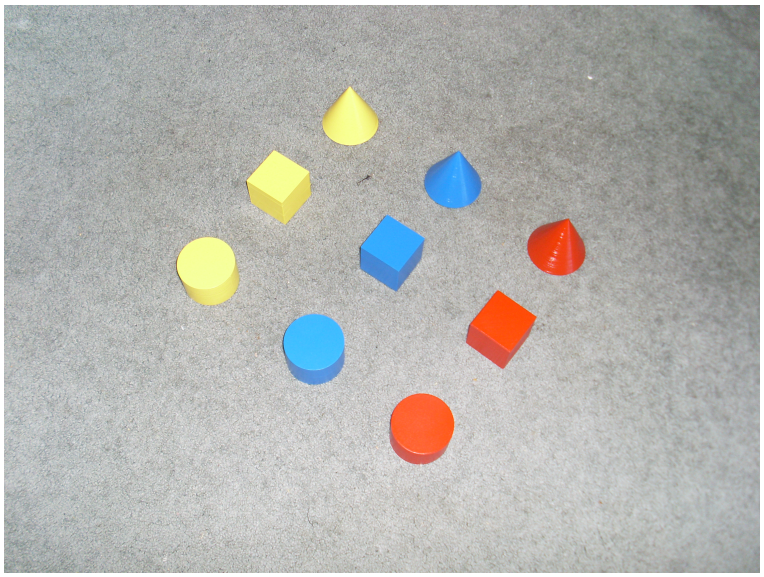

Figure 1: Test Objects

We used the following angles/distances for capturing the objects:

Z Angle	Orthogonal Distance To Object	Height Offset
0	18-32 in	0
45	18-32 in	0
90	18-32 in	0
135	18-32 in	0
180	18-32 in	0
225	18-32 in	0
270	18-32 in	0
315	18-32 in	0
0	18-32 in	10 in
45	18-32 in	10 in
90	18-32 in	10 in
135	18-32 in	10 in
180	18-32 in	10 in
225	18-32 in	10 in
270	18-32 in	10 in
315	18-32 in	10 in
0	0	36-48 in

The camera we used to capture the images was a Samsung L730. We used auto-focus and disabled flash.

We took pictures of the train objects in six diverse locations, four indoor locations with a variety of backgrounds (carpet, hardwood floor, vinyl, concrete) and two outdoor locations (concrete and grass). The outdoor pictures were taken in the early afternoon.

4.2 Models

Selecting models, as with most deep-learning tasks, involved some level of trial and error. Since we wanted to explore the efficacy of transfer learning for this task, we picked a variety of pre-existing image classification networks to see if any would perform better for the task:

1. Xception [2]
2. VGG16 [3]
3. VGG19 [3]
4. ResNet50 [4]
5. InceptionV3 [5]
6. InceptionResNetV2 [6]
7. MobileNet [7]
8. DenseNet [8]

We also decided to peel off several layers of the networks to gain access to higher-dimension feature spaces which might be necessary for such a complicated task. We tried to pick layers strategically, such as after a block of the model's calculations:

1. Xception: {0, 8}
2. VGG16: {0, 3, 8}
3. VGG19: {0, 3, 9}
4. ResNet50: {0, 3, 13}
5. InceptionV3: {0, 2, 19}

6. InceptionResNetV2: {0, 5, 20}
7. MobileNet: {0, 13, 20}
8. DenseNet: {0, 3, 10}

Importantly, before inputting the data into our own model, we applied a standard scalar transformation (trained on the output of the training data from the transfer model), then a normalization operation. Our reasoning was that such operations seemed to reduce the chance that the model would learn the all 0's classifier.

For our own model, we opted for a densely-connected network. Our reasoning behind that decision was that, since the transfer models used convolutional layers, much of the image information will have already been processed at the point where our model is used. Thus, our model did not need convolutional layers.

We varied the number of dense layers between 1, 3 and 4 and the number of neurons per layer between 500, 1000, and 1500. Experimentally, this seemed like a good balance, as adding more layers and neurons tended to make our network learn the all 0's classifier.

To see if there was a point at which the test data reached a maximum accuracy, we trained each generated network for 10 epochs.

Additional Settings:

- Optimization Algorithm: Adam. The reason for this decision is that Adam is an improvement over SGD for this type of application. From the literature, it sounds that Adam also has some automatic parameter estimation, as well.
- Loss Function: Binary Crossentropy. This is necessary for multi-label classification
- Last Layer Activations: Sigmoid. This is necessary for multi-label classification

5 Conclusion

Our findings were that many model configurations descended to the all-0's output model. We observed that, as the number of neurons per layer and number of densely connected layers in our network increased, the probability of descending to the all-0's output model approached 1. However, some model configurations were able to improve accuracy on the training data, but no model was able to find any connection between the training data and testing data; as the training accuracy increased, test accuracy was inversely affected.

One possibility for this outcome is that we didn't collect enough data to allow the network to learn the abstractions of the various characteristics. Although the network was able to improve on the training set some, the network was unable to apply this learning to any of the test sets.

Another possibility is that such a complicated task requires a specially-built convolutional neural network. Whereas we used a transfer learning approach, image classification and object characteristic enumeration could be so different that we cannot transfer the knowledge of image classification networks to the task of object characteristic enumeration.

Unfortunately, as neural networks are a black-box model, it is difficult to know for certain the reason that our approach did not work, although the presentation of a working model may shed light onto the task (see below).

5.1 Results

Selected results are below. Note that none performed remarkably better than the all 0's classifier.

Figures 5 and 6 below illustrate the typical behavior of many of our generated models.

Transfer Model	Transfer Model Top Layer	Our Model # Layers	Our Model Neurons/Layer	Tr Best Epoch	Tr Best Accur	Te_1 Best Epoch	Te_1 Best Accur
inception_v3	average_pooling2d_9	1	500	10	1	2	0.668845336811215
vgg19	block4_pool	1	500	10	0.999128536149567	3	0.66884533057805
densenet	fc1000	1	500	10	0.7812636229727	7	0.668845330188477
mobilenet	conv_pw_11_relu	3	1500	10	0.999128538642834	2	0.668845330188477
mobilenet	conv_pw_12_relu	4	1500	7	0.987363828553094	2	0.668845317722146

Figure 2: The model configurations with the best Te_1 accuracy

Transfer Model	Transfer Model Top Layer	Our Model # Layers	Our Model Neurons/Layer	Tr Best Epoch	Tr Best Accur	Te_2 Best Epoch	Te_2 Best Accur
vgg19	flatten	1	500	10	0.985185171731936	3	0.839999997615814
inception_resnet_v2	predictions	1	500	10	0.733115470643137	6	0.839999992847443
inception_resnet_v2	predictions	1	1500	10	0.808278853986777	4	0.839999992847443
xception	predictions	1	1500	10	0.828758158948686	1	0.836666655540466
xception	predictions	3	500	10	0.953376903565101	3	0.836666655540466

Figure 3: The model configurations with the best Te_2 accuracy

Transfer Model	Transfer Model Top Layer	Our Model # Layers	Our Model Neurons/Layer	Tr Best Epoch	Tr Best Accur	Te_3 Best Epoch	Te_3 Best Accur
vgg19	predictions	3	1500	9	0.722657944330203	6	0.866666674613953
inception_resnet_v2	block8_9_ac	1	500	9	1	4	0.866666674613953
inception_v3	predictions	1	1000	10	0.837908495641222	8	0.85833340883255
inception_resnet_v2	block8_9_ac	1	1500	7	1	3	0.85833340883255
vgg19	predictions	4	500	10	0.729193900691138	8	0.858333289623261

Figure 4: The model configurations with the best Te_3 accuracy

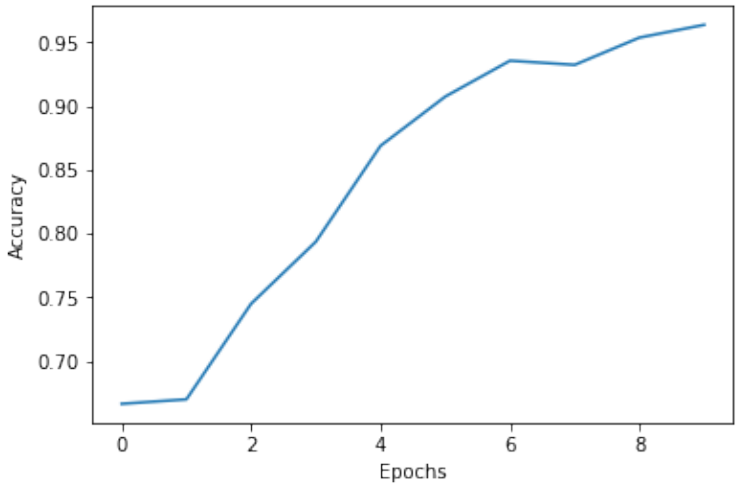

Figure 5: Tr accuracy for the model using "Xception," 0 peeled layers, with our model having 4 dense layers and 2500 neurons per layer

5.2 Recommended Further Work

5.2.1 Experimenting with More Training and Test Data

Though nearly a thousand pictures were taken, the objects were only captured from several angles (17) and in several locations (6). This could have been detrimental to the model, as, without enough training examples, the model may not learn to abstract the characteristics of the objects. We recommend that further work augment the data more. One method would be to use a webcam and capture a video of each object from a full 360-degree arc. Another method would be to use more locations, or even capture an object as it is moving through locations, for example, being held in a hand as the person holding the object walks around.

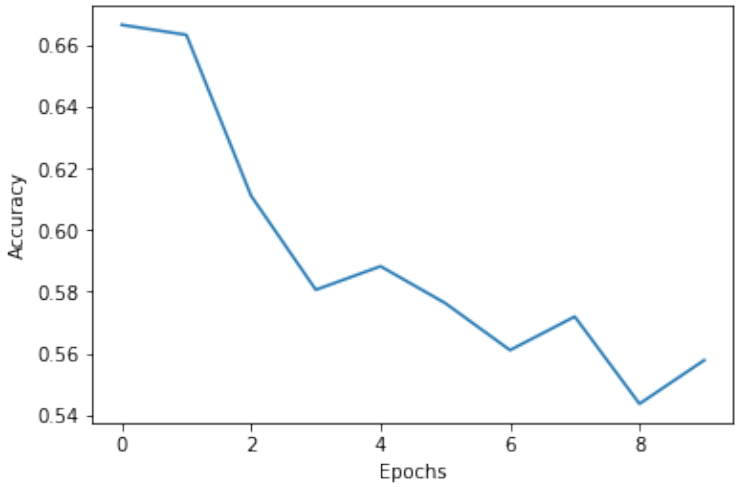

Figure 6: Te_1 accuracy for the model using "Xception," 0 peeled layers, with our model having 4 dense layers and 2500 neurons per layer

5.2.2 Create a CNN Specially Optimized for this Task

Though we opted to make use of existing networks, this may have been a hindrance rather than a benefit. Such a complicated task may demand a network that is optimized for the task, rather than transfer learning from the output of an image classification network. An experiment with a carefully-constructed convolutional neural network may bring insight as to whether transfer learning is helpful or necessary for this task.

5.2.3 Embodying the Model in a Robotic Agent

The original motivation of this work was to, eventually, incorporate capturing an object from a variety of angles into a robotic agent to allow it to learn in an interactive learning session with a human. However, improvements to the model should be made before attempting this.

References

[1] Ian Goodfellow, Yoshua Bengio, Aaron Courville. *Deep Learning*, The MIT Press, Cambridge, MA, 1st Ed., 2016.

[2] Franois Chollet. "Xception: Deep Learning with Depthwise Separable Convolutions," Cornell University Library, 2016. https://arxiv.org/abs/1610.02357

[3] Karen Simonyan, Andrew Zisserman. "Very Deep Convolutional Networks for Large-Scale Image Recognition," Cornell University Library, 2014. https://arxiv.org/abs/1409.1556

[4] Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun. "Deep Residual Learning for Image Recognition," Cornell University Library, 2015. https://arxiv.org/abs/1512.03385

[5] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, Zbigniew Wojna. "Rethinking the Inception Architecture for Computer Vision," Cornell University Library, 2015. https://arxiv.org/abs/1512.00567

[6] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, Alex Alemi. "Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning," Cornell University Library, 2016. https://arxiv.org/abs/1602.07261

[7] Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, Hartwig Adam. "MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications," Google, Inc. https://arxiv.org/pdf/1704.04861.pdf

[8] Gao Huang, Zhuang Liu, Laurens van der Maaten, Kilian Q. Weinberger. "Densely Connected Convolutional Networks," Cornell University Library, 2016. https://arxiv.org/abs/1608.06993