Contrastive Model

Created by Unknown User (barrelchester), last modified on Apr 14, 2023

Again, I will be comparing model training and test results on **linear** RGB images versus **log** RGB images (linearization applied first, then log).

This approach uses a contrastive model based on CLCC. The idea is to learn a model which encourages image vector representations with the same illuminant to be close and the same image with different illuminants to be far from each other. This teaches the network to differentiate the semantics of the image (which the pretrained backbone model is tuned for) from the illuminant features of the image.

The objective is to minimize the following metric:

value = dist(anchor, pos) - dist(anchor, neg) + margin

where "dist" is some vector distance function. e.g. cosine

"anchor" is a training image

"pos" is another training image with the anchor's illuminant applied to it

"neg" is the anchor with a random illuminant applied to it

"margin" is the amount by which we would like the two vectors to differ

Training Data and Pretrained Backbone Model

Again, the SimpleCube dataset is used.

I'm experimenting with **MobileNetV2** (3.5M params) and **GoogLeNet** (6.6M params) as backbone models.

First, an image embedding model ("**contrastive embedding model**") is trained using the triplets described above, and then an RGB regression output layer is added to predict the illuminant and finetuned ("**illuminant prediction model**").

Contrastive Embedding Model

This model is trained just to appropriately embed images with respect to illuminant, so there are no test metrics to report.

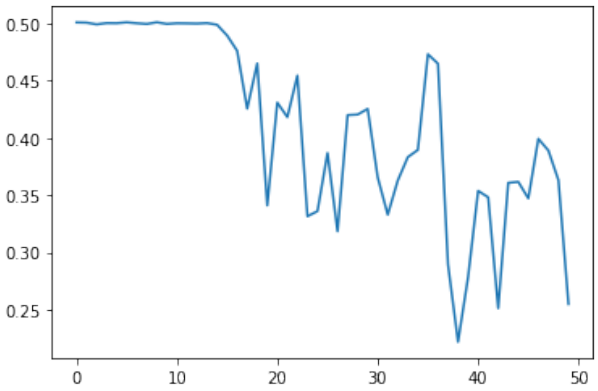
Example of an anchor, positive, and negative triplet:

anchor positive negative

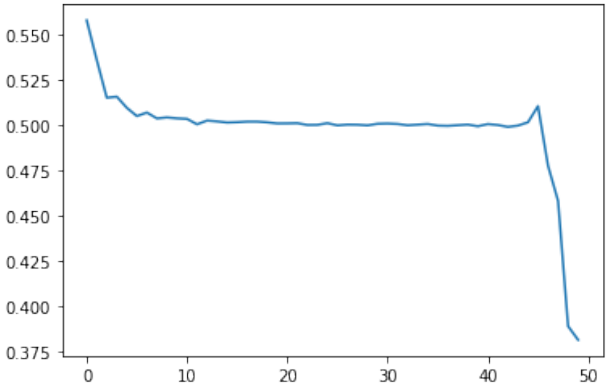
GoogLeNet

learning rate 1e-5, distance function = cosine

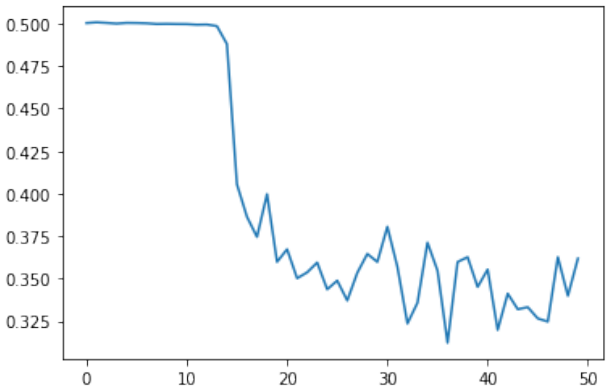
Linear Training Curve



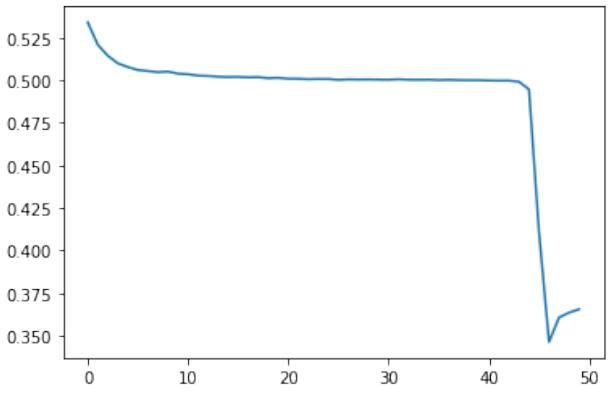
Log Training Curve



Linear Cross Validation Curve



Log Cross Validation Curve



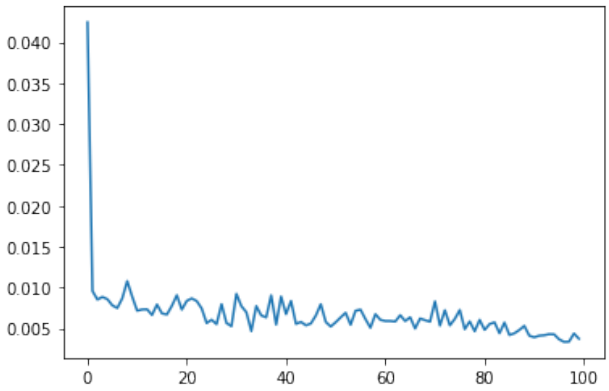
Illuminant Prediction Model

The learned parameters are copied from the embedding model into a new model with an RGB regression output layer added. Either the whole model can be trained or just the output layer (embedding layers are **frozen**). After training, the test data is run through the model and the test metrics are calculated.

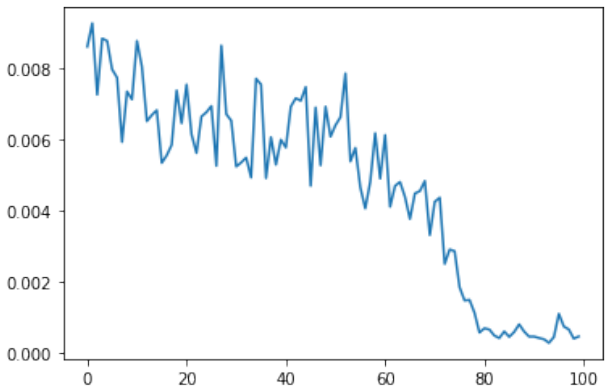
MobileNetV2

learning rate 1e-3

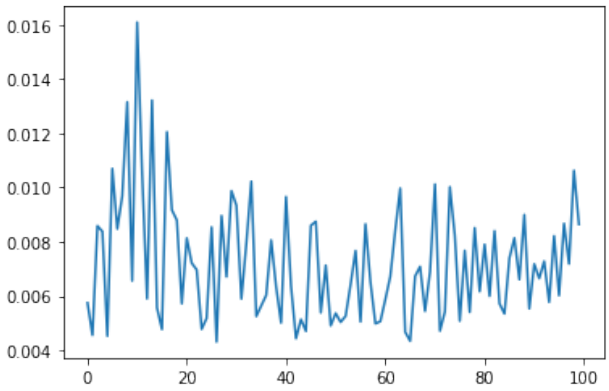
Linear Training Curve



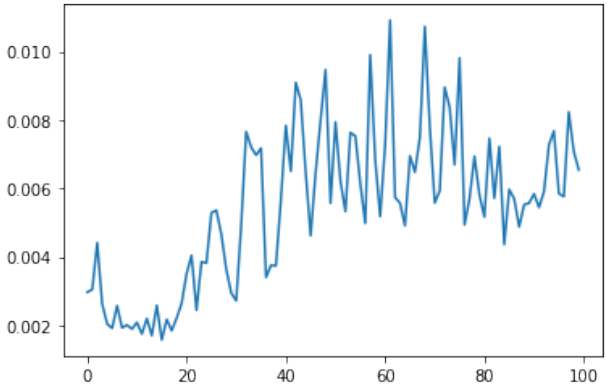
Log Training Curve



Linear Cross Validation Curve



Log Cross Validation Curve



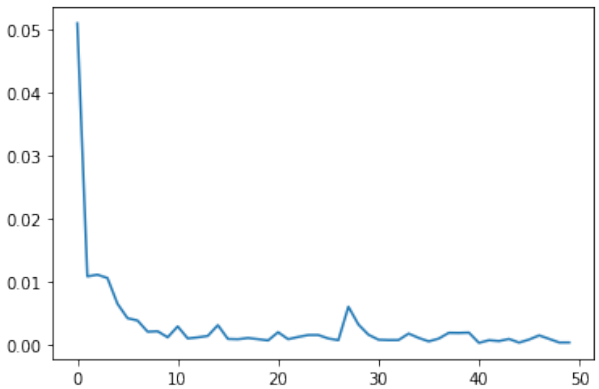
Test Metrics

Type	Epochs	Mean	Median	Trimean	Worst25	Best25
contrastive, moblenet lin	100	4.210256	1.641216	3.682925	1.076623	11.794701
contrastive, moblenet lin-log	100	4.739438	3.630542	4.426092	0.953732	10.550815

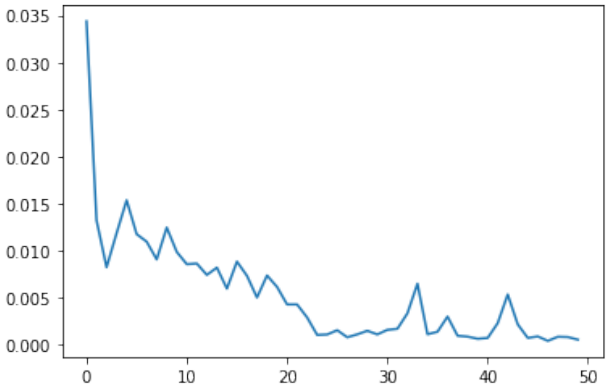
GoogLeNet

learning rate 1e-4

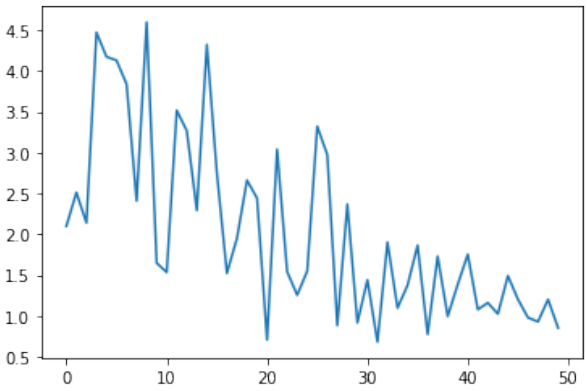
Linear Training Curve



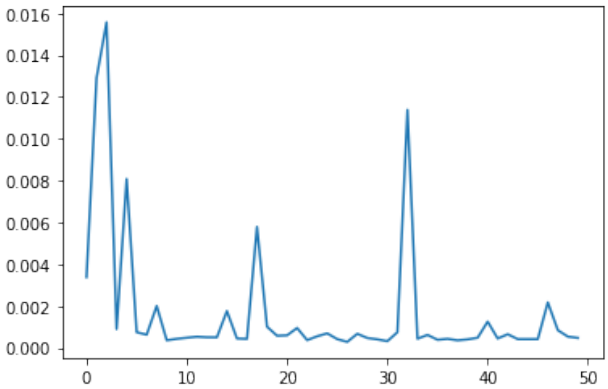
Log Training Curve



Linear Cross Validation Curve



Log Cross Validation Curve



Test Metrics

Type	Epochs	Mean	Median	Trimean	Worst25	Best25
contrastive, googlenet lin	50	2.978169	2.170369	2.690430	0.740337	6.757335
contrastive, googlenet lin-log	50	2.194672	1.475897	2.004171	0.712219	4.805640
contrastive, googlenet lin-log	100	1.992313	1.587882	1.831709	0.522633	4.273091

Results Summary

Best scores from the non-contrastive models:

Type	Epochs	Mean	Median	Trimean	Worst25	Best25
googlenet lin, expand, log	200	1.420396	0.869077	1.242575	0.286096	3.530126

Best scores from the contrastive models:

Type	Epochs	Mean	Median	Trimean	Worst25	Best25
contrastive, googlenet lin-log	100	1.992313	1.587882	1.831709	0.522633	4.273091

No labels