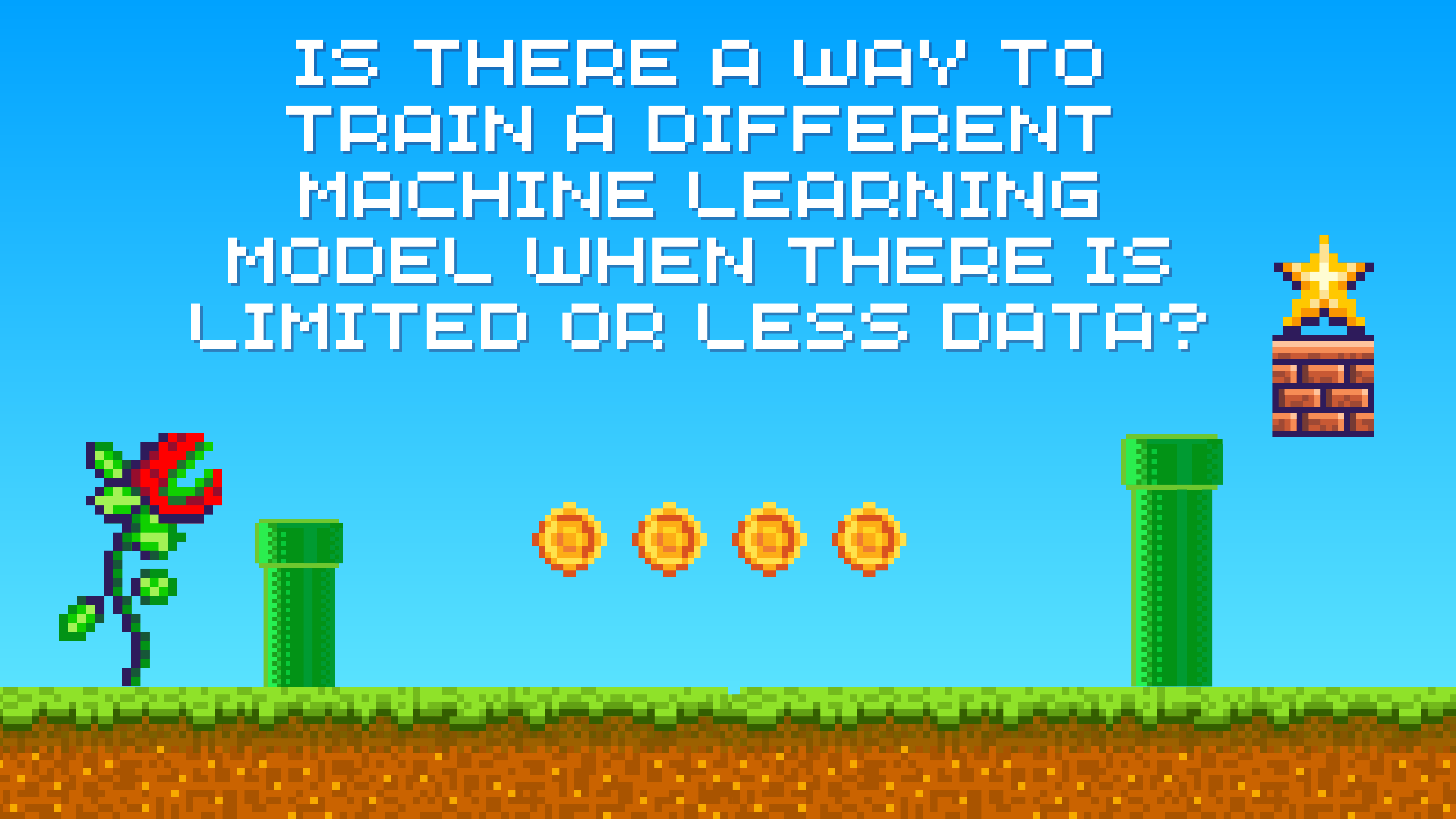
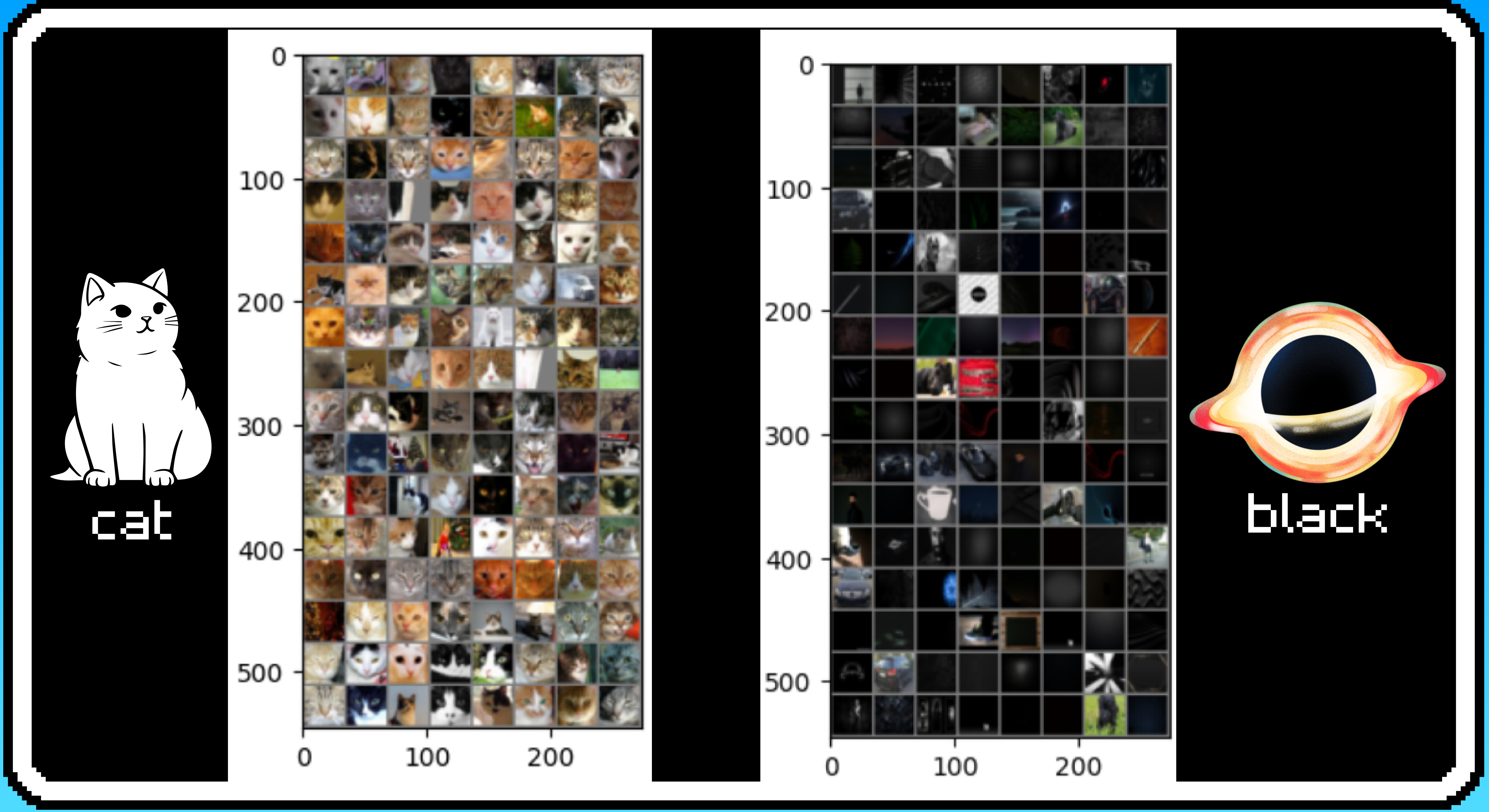
GENERATING MACHINE LEARNING MODELS USING MACHINE LEARNING MODELS

Aditya Patel
Karan Jain

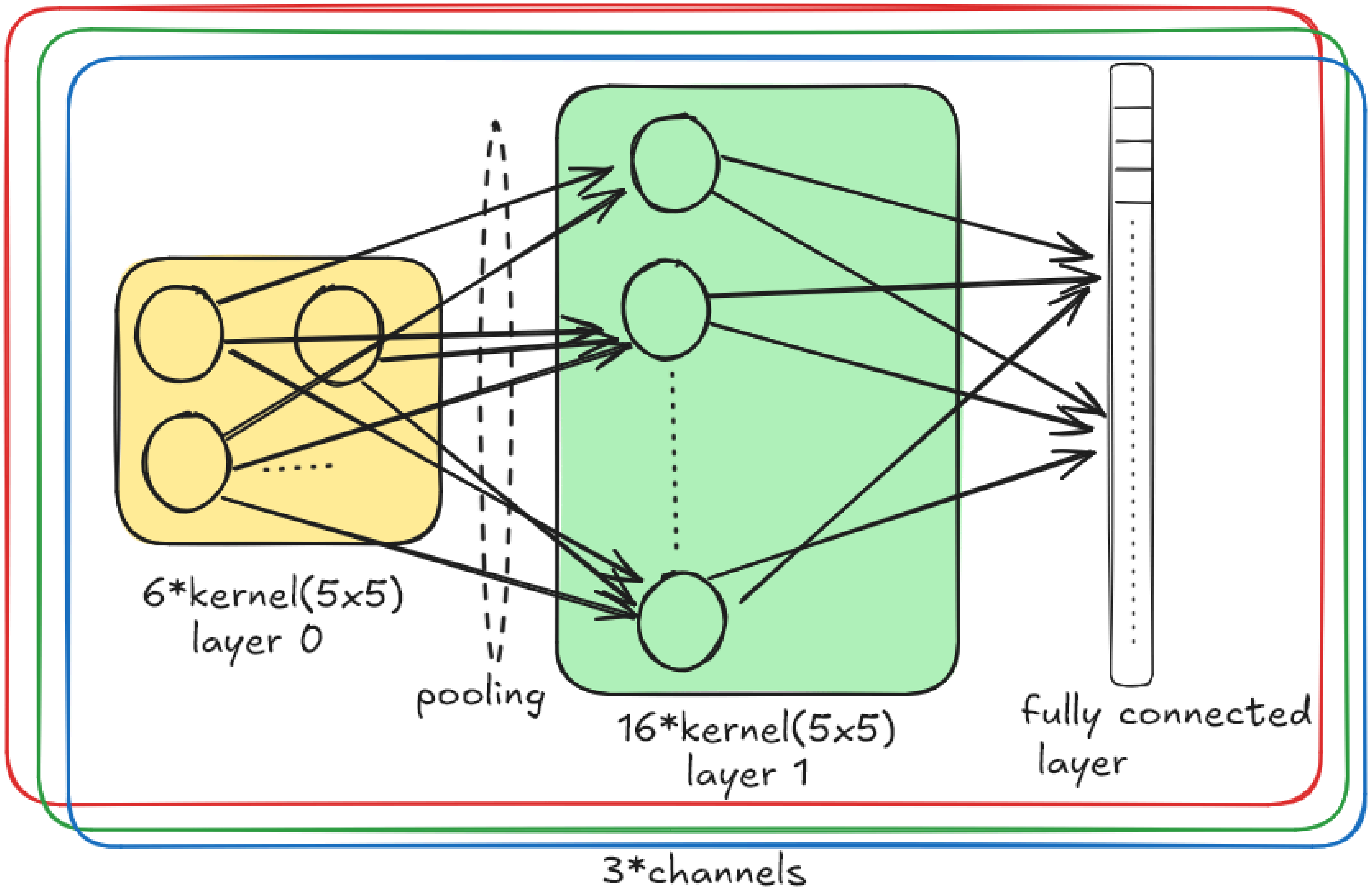

IS THERE A WAY TO
TRAIN A DIFFERENT
MACHINE LEARNING
MODEL WHEN THERE IS
LIMITED OR LESS DATA?

- Generate a neural network without actually training the neural network!
- Combine two or more neural networks together to produce a third
- We trained a CNN to detect cats using a dataset of 29K cats, and then we trained another CNN to detect black color using a dataset 1.7k images.

WHAT??


cat
0
100
200
300
400
500
0
100
200
0
100
200
300
400
500
0
100
200
black

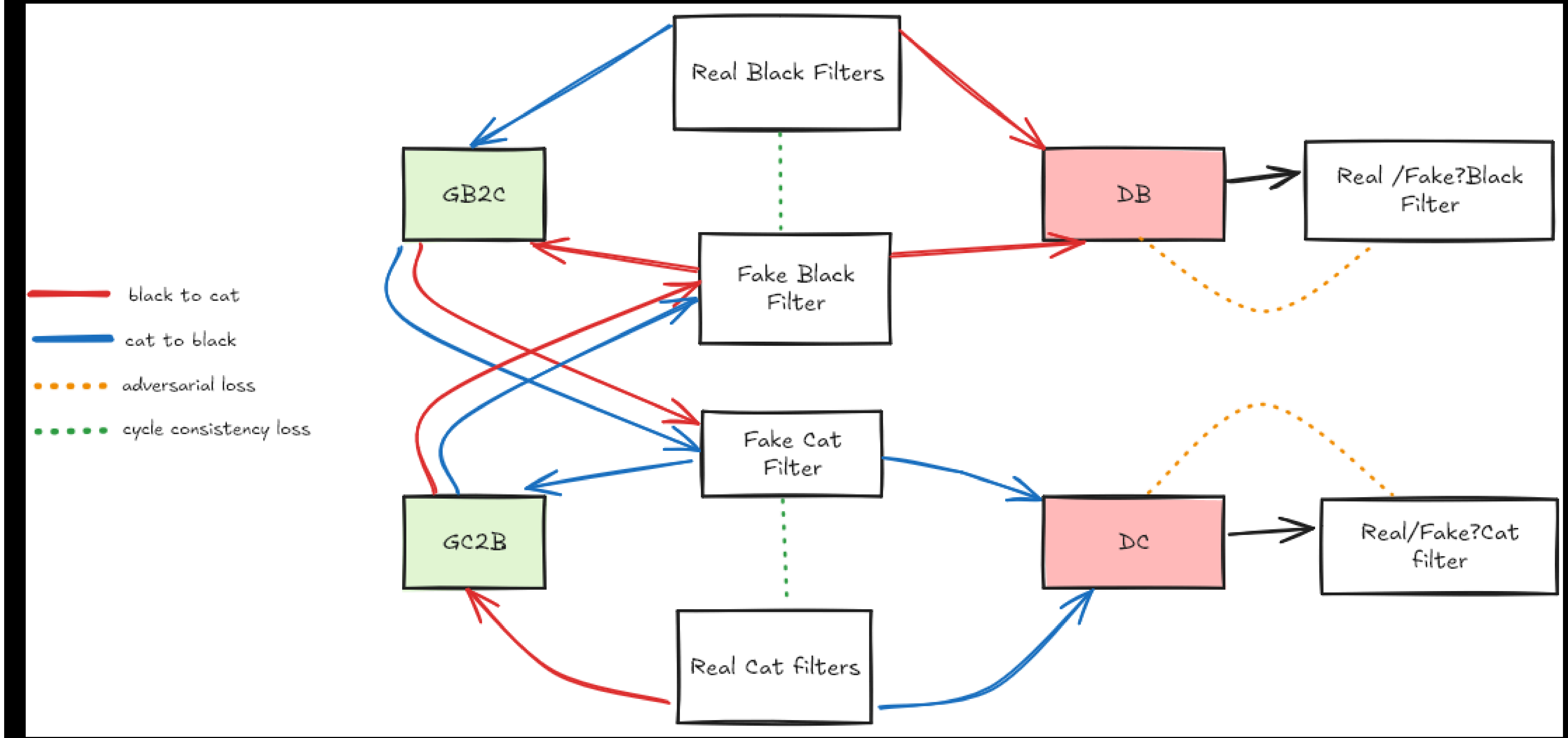
Dataset


6*kernel(5x5)
layer 0
pooling
16*kernel(5x5)
layer 1
fully connected
layer
3*channels

Base CNN

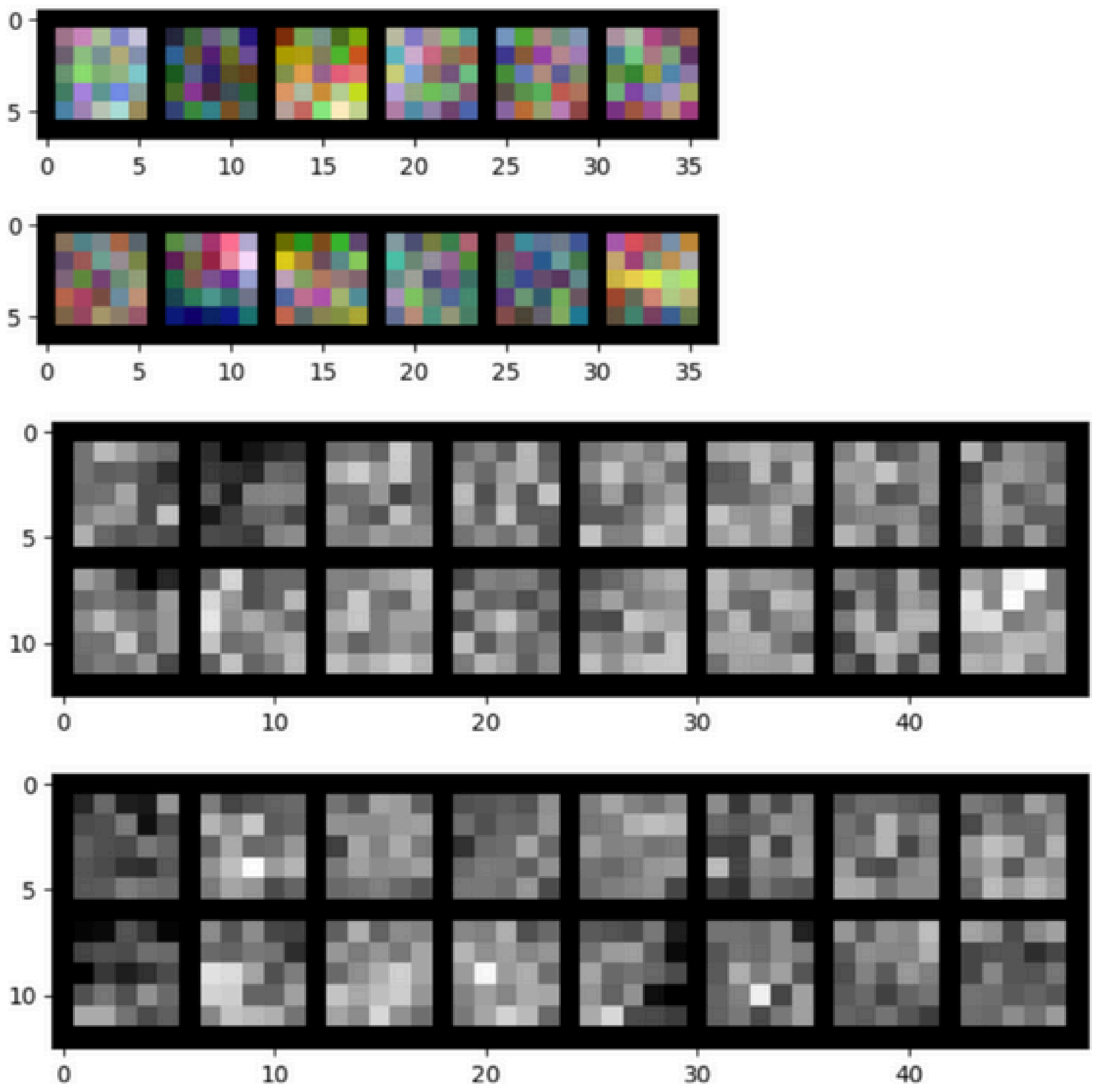
- We then trained a CycleGAN over the filters of these 2 CNN's
- The CycleGAN learned to create new "images" (filters) that combine the characteristics of both CNNs (the generator(s) tries to "fool" the discriminator by producing fake black CNN filters and fake cat CNN filters)

HOW??

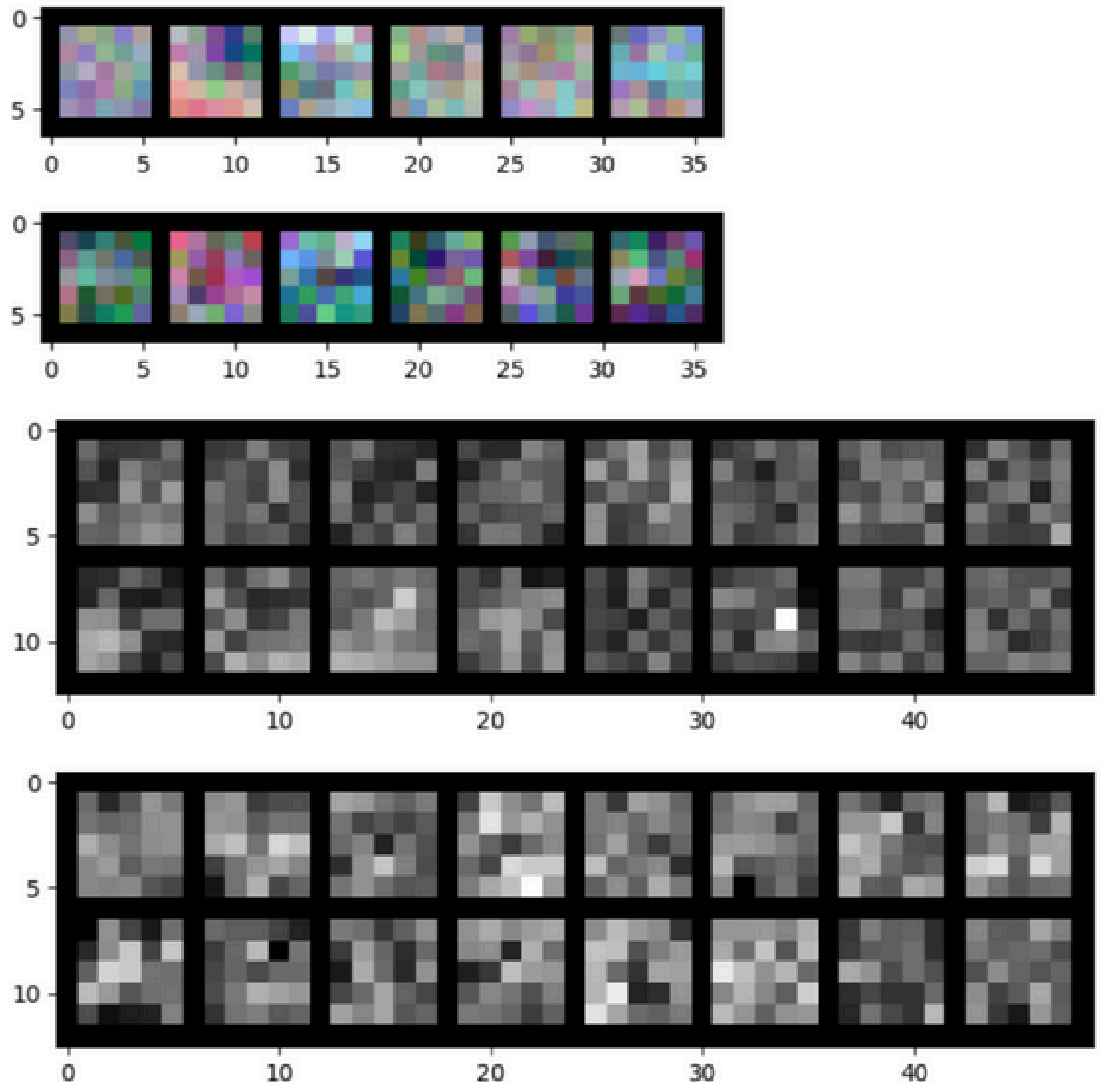

Real Black Filters
GB2C
DB
Real /Fake?Black Filter
Fake Black Filter
black to cat
cat to black
adversarial loss
cycle consistency loss
Fake Cat Filter
GC2B
DC
Real/Fake?Cat filter
Real Cat filters

CycleGAN

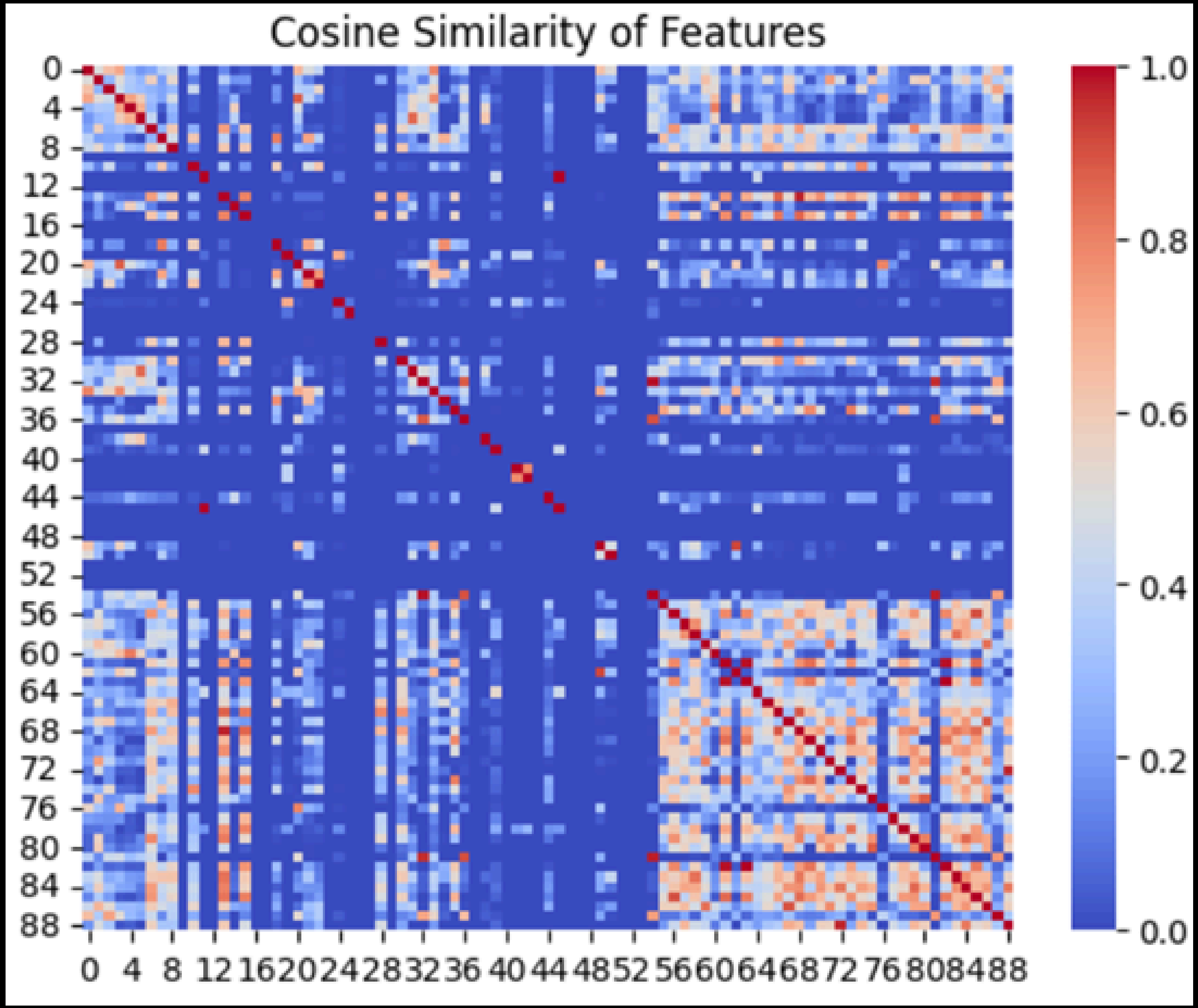
Generated CNN



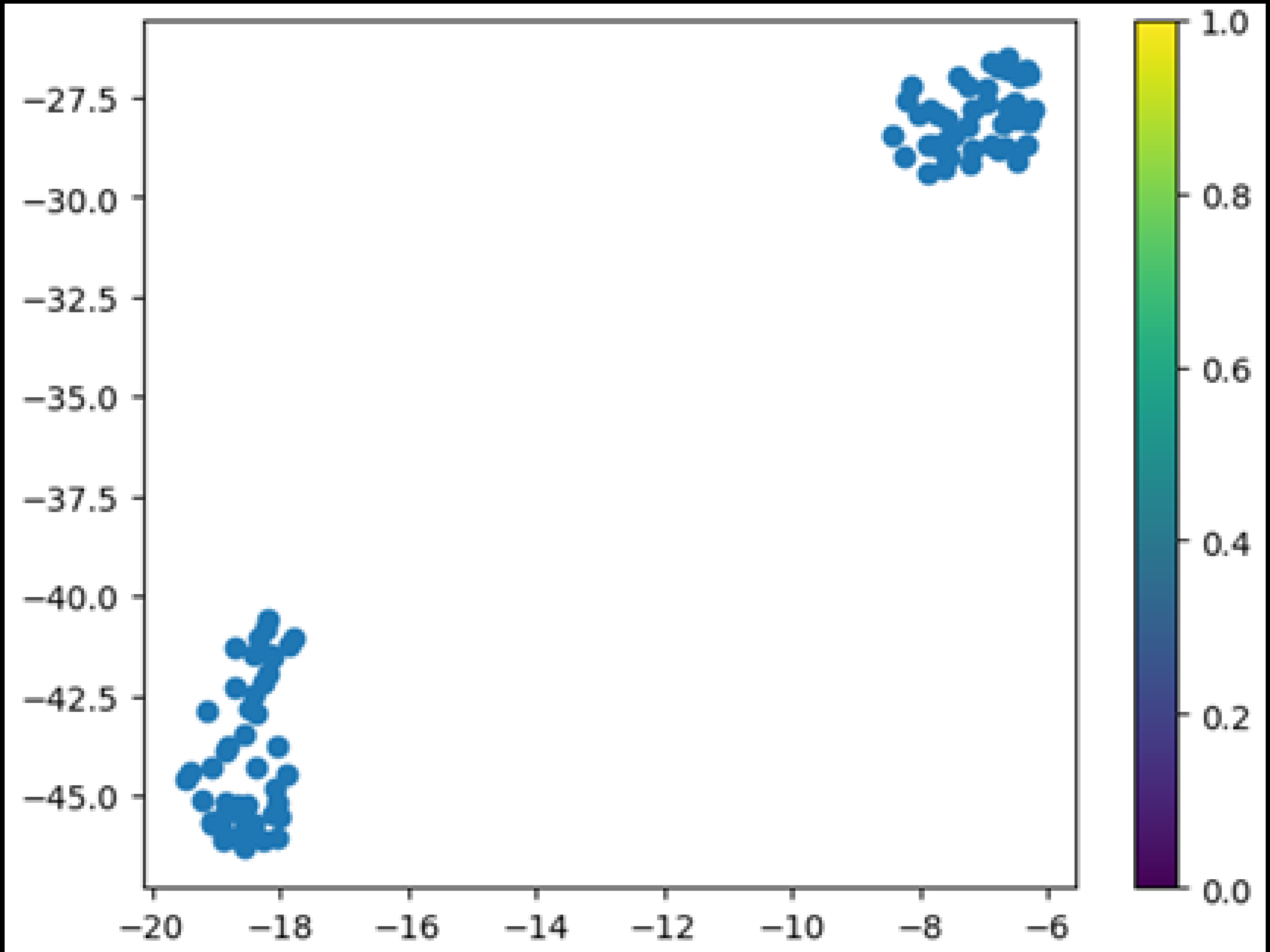
Kernels of CNN Model A and the kernels generated from them using CycleGAN



Kernels of CNN Model B and the kernels generated from them using CycleGAN

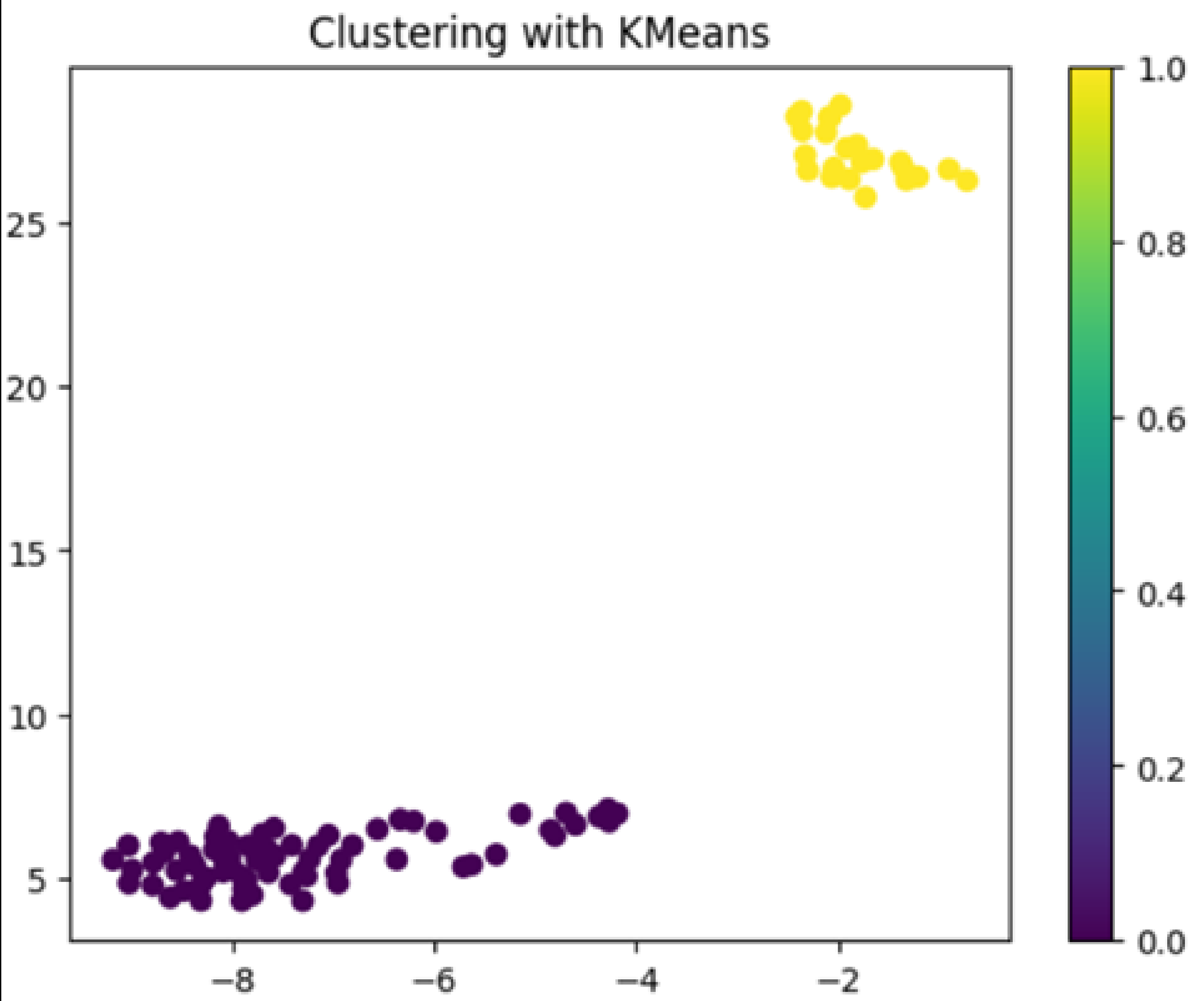


Cosine Similarity

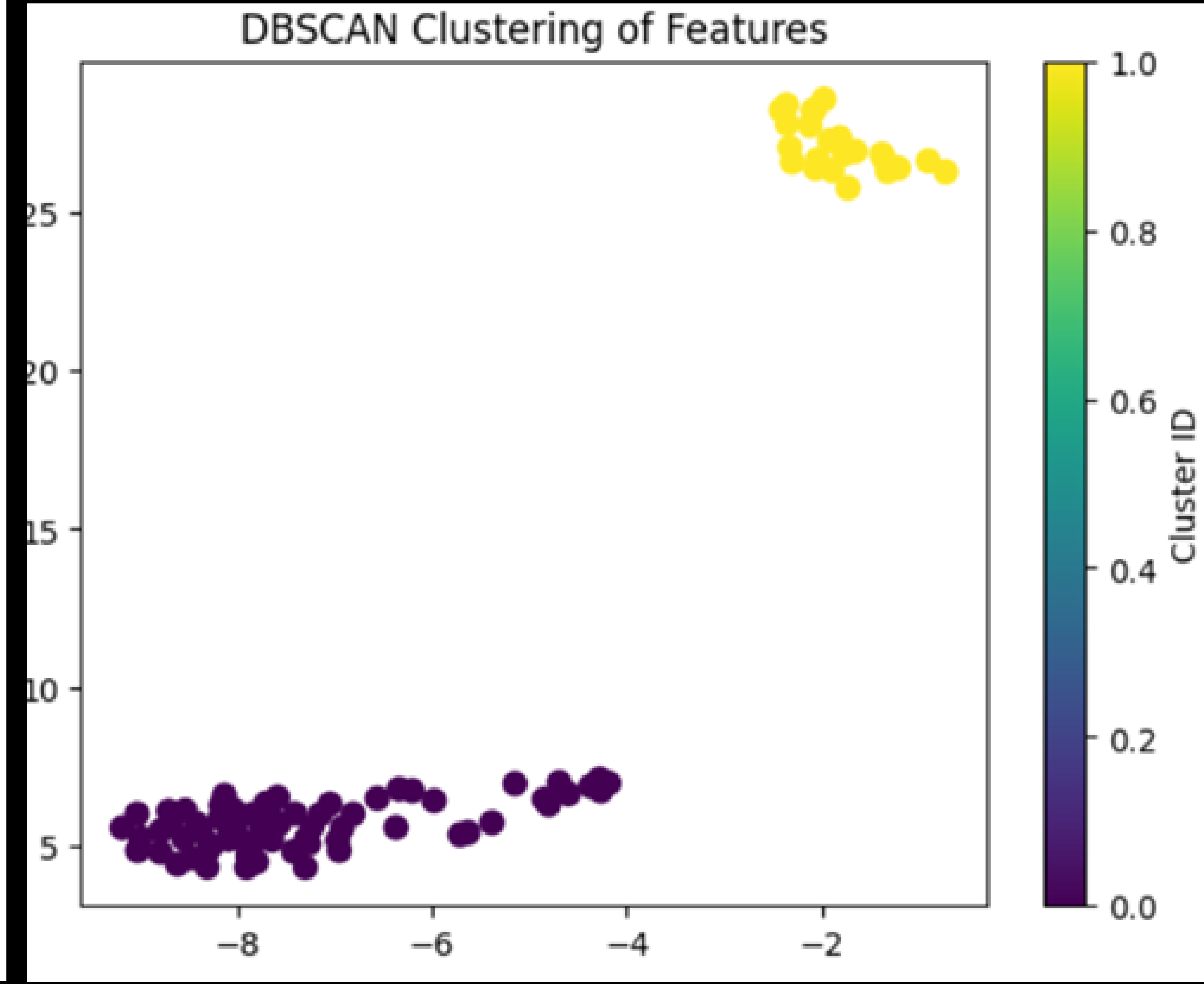


−27.5
−30.0
−32.5
−35.0
−37.5
−40.0
−42.5
−45.0
−20
−18
−16
−14
−12
−10
−8
−6
1.0
0.8
0.6
0.4
0.2
0.0

UMAP



K-Means



DBScan

Model	Generated Filter Domain	Accuracy	Precision	Recall	Entropy	Purity
229	A	0.876	0.852	0.828	0.537	0.876
1486	A	0.876	0.823	0.848	0.528	0.876
1234	A	0.842	0.911	0.738	0.61	0.842
1330	A	0.842	0.735	0.833	0.59	0.842
1304	A	0.82	0.852	0.725	0.672	0.82
1314	A	0.82	0.9411	0.695	0.629	0.82
226	A	0.808	0.588	0.869	0.562	0.809
307	A	0.797	0.558	0.863	0.566	0.797
229	B	0.797	0.823	0.7	0.724	0.797
478	B	0.797	0.588	0.833	0.605	0.797
805	B	0.797	0.764	0.722	0.723	0.797
1291	B	0.797	0.676	0.766	0.686	0.797
1205	A	0.786	0.617	0.777	0.673	0.786
1567	A	0.786	0.617	0.777	0.673	0.786
311	B	0.786	0.647	0.758	0.697	0.786
253	A	0.775	0.705	0.705	0.756	0.775
1215	B	0.764	0.882	0.638	0.75	0.764
1278	A	0.752	0.911	0.62	0.739	0.752
1703	A	0.752	0.705	0.666	0.801	0.752
214	A	0.741	0.676	0.657	0.814	0.741
959	A	0.741	0.529	0.72	0.72	0.741
282	B	0.741	0.794	0.627	0.817	0.741
1470	A	0.73	0.735	0.625	0.84	0.73
1507	A	0.73	0.794	0.613	0.831	0.73
1311	A	0.696	0.823	0.571	0.849	0.696
1337	A	0.696	0.911	0.563	0.775	0.696
1540	B	0.606	0.735	0.49	0.935	0.606

K-Means

Model	Generated Filter Domain	Accuracy	Precision	Recall	Entropy	Purity
229	A	0.887	0.852	0.852	0.501	0.887
1486	A	0.887	0.823	0.875	0.568	0.876
282	B	0.865	0.764	0.866	1.926	0.573
229	B	0.853	0.705	0.888	1.281	0.707
1234	A	0.842	0.911	0.738	0.61	0.842
1330	A	0.842	0.705	0.857	0.698	0.831
1470	A	0.842	0.647	0.916	1.5	0.606
1540	B	0.842	0.647	0.916	1.793	0.483
1314	A	0.831	0.882	0.731	1.396	0.662
1304	A	0.82	0.852	0.725	0.794	0.808
226	A	0.808	0.588	0.869	0.562	0.809
478	B	0.797	0.588	0.833	1.178	0.617
805	B	0.797	0.764	0.722	0.723	0.797
307	A	0.786	0.529	0.857	0.569	0.786
1567	A	0.786	0.617	0.777	0.673	0.786
1703	A	0.786	0.676	0.741	1.268	0.707
311	B	0.786	0.647	0.758	1.292	0.528
1291	B	0.786	0.588	0.8	1.62	0.573
253	A	0.775	0.705	0.705	0.756	0.775
1507	A	0.775	0.529	0.818	1.737	0.5584
214	A	0.764	0.676	0.696	0.769	0.764
1205	A	0.752	0.5	0.772	0.653	0.752
959	A	0.741	0.529	0.72	0.781	0.741
1311	A	0.719	0.617	0.636	1.777	0.539
1337	A	0.696	0.911	0.563	0.877	0.685
1278	A	0.674	0.676	0.56	1.818	0.46
1215	B	0.606	0.294	0.476	2.017	0.449

DBScan

Model	Generated Filter Domain	Accuracy	Precision	Recall	Entropy	Purity
1304	A	0.82	0.882	0.714	0.667	0.82
1314	A	0.764	0.852	0.644	0.767	0.764
229	A	0.752	0.911	0.62	0.739	0.752
1337	A	0.741	0.941	0.603	0.716	0.741
1486	A	0.741	0.941	0.603	0.716	0.741
1234	A	0.73	0.911	0.596	0.757	0.73
1278	A	0.73	0.911	0.596	0.757	0.73
1507	A	0.719	0.9411	0.581	0.729	0.719
1205	A	0.707	0.911	0.574	0.7704	0.707
282	B	0.685	0.911	0.553	0.778	0.685
1291	B	0.685	0.911	0.553	0.778	0.685
1311	A	0.64	0.911	0.516	0.781	0.674
253	A	0.629	0.911	0.508	0.778	0.685
805	B	0.5955	0.9411	0.484	0.716	0.741
1215	B	0.55	0.911	0.455	0.728	0.764
1703	A	0.505	0.941	0.432	0.61	0.831
307	A	0.494	0.9411	0.426	0.591	0.842
214	A	0.449	0.911	0.402	0.561	0.865
959	A	0.426	0.882	0.389	0.569	0.865
1540	B	0.426	0.882	0.389	0.569	0.865
311	B	0.415	0.911	0.387	0.471	0.898
229	B	0.404	0.941	0.385	0.355	0.932
1470	A	0.382	0.941	0.376	0.262	0.955
226	A	0.359	0.882	0.361	0.338	0.932
1330	A	0.359	0.911	0.364	0.245	0.955
1567	A	0.359	0.911	0.364	0.245	0.955
478	B	0.359	0.911	0.364	0.245	0.955

One-Class SVM

Discussion

We find that the black cat images were clustered together effectively demonstrating unsupervised learning, i.e. the generated CNN was able to differentiate black cat images from others without being trained on any data....

A few challenges observed during the project were that of hyperparameter selection, dimensionality of the feature vectors and the translation of filters from one domain to another

ITS'A POSSIBLE!!

Future Work

- Hyper-parameter Optimization for Clustering
- Feature space exploration with alternative metrics and feature weighting
- Semi Supervised Learning with small dataset
- Improving domain translation
- End to end pipeline development based on semantics

Conclusion

THANK YOU!

END