Deep similarity learning of medical images to support explainable artificial intelligence classification

M.Sc. Artificial Intelligence and Data Engineering

Candidate
Federico Minniti

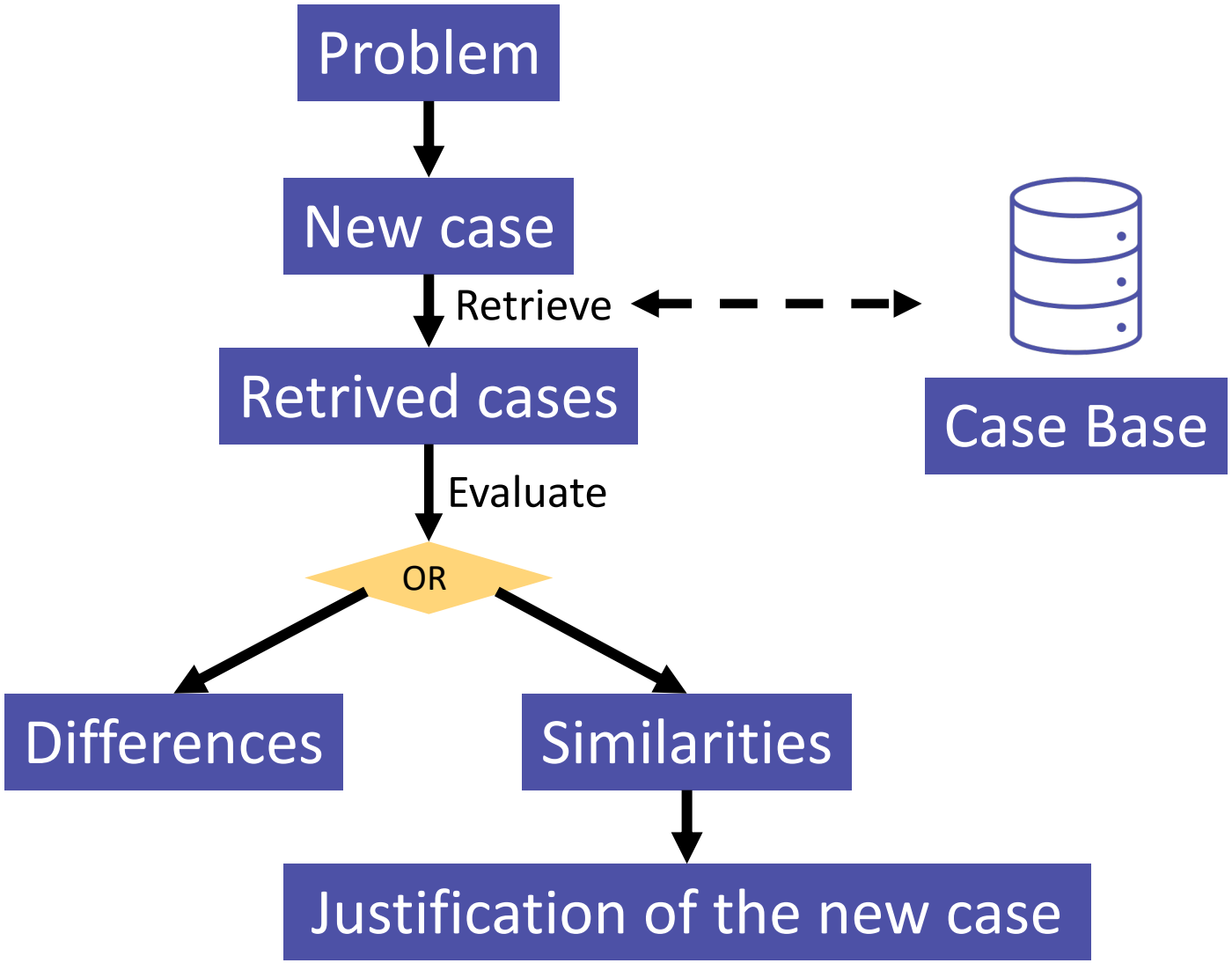
Supervisors
Mario G.C.A. Cimino
Federico A. Galatolo
Marco Parola

UNIVERSITÀ DI PISA

Problem introduction

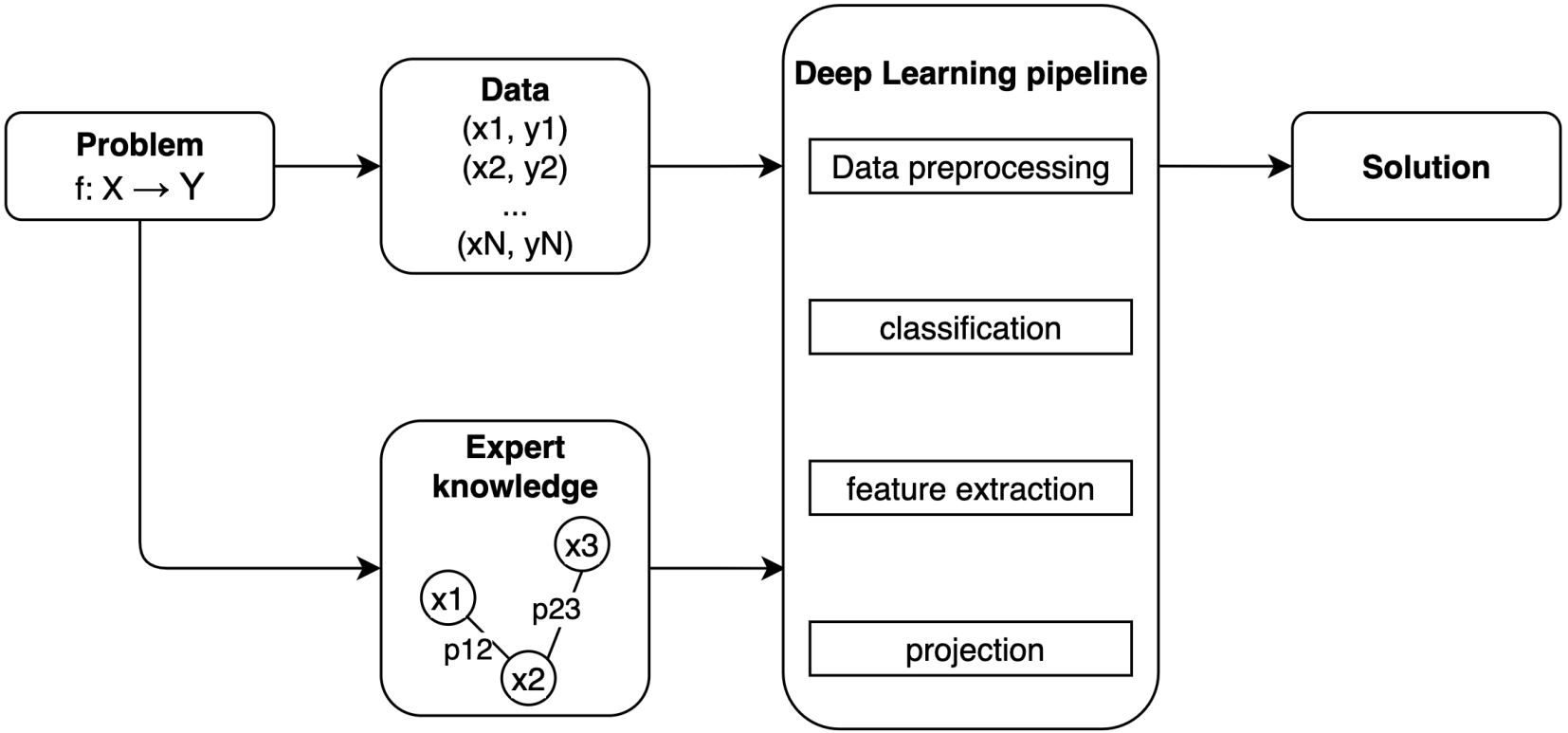
- Oral cancer is the 7th most frequent cancer and the 9th cause of death worldwide, with approximately 710,000 cases and 359,000 deaths per year.
- AI is generally considered like a black box
  - Critical applications (eg: medical field)
  - Explainable artificial intelligence
- To explain our system we will exploit:
  - Case base reasoning
  - Informed Deep Learning.
- Problems:
  - Classify oral cancer cases (neoplastic, traumatic and aphthous)
  - Explain the classifier's prediction exploring the features space

Case Base Reasoning (interpretive system)



Xu, L.D., 1994. Case based reasoning. IEEE potentials 13, 10–13.

Informed Deep Learning (IDL)



Von Rueden, L., Mayer, S., Beckh, K., Georgiev, B., Giesselbach, S., Heese, R., Kirsch, B., Pfrommer, J., Pick, A., Ramamurthy, R., et al., 2021. Informed machine learning–a taxonomy and survey of integrating prior knowledge into learning systems. IEEE Transactions on Knowledge and Data Engineering 35, 614–633.

Classification models, autoencoders and triplet network

- Regarding classification models we exploit some well knwon and utilized models in literature[1][2]:
  - ResNet, GoogLeNet, ViT, EfficientNet, DenseNet, Swin, VGG, RegNet, MobileNet, MaxVit, ConvNeXt
- We also developed Autoencoder and Variational Autoencoder
- Regarding the triplet net:
  - As loss function we exploit the triplet loss deeply inspected in literature[3]
  - We developed our own version varying the number of linear layers (2,3,5)
- Evaluation metrics: Jaccard distance, Normalized Spearman footrule distance, Normalized Kendall tau distance, Compound value

[1] Singha Deo, B., Pal, M., Panigrahi, P.K., Pradhan, A., 2022. Supremacy of attention based convolution neural network in classification of oral cancer using histopathological images. medRxiv, 2022–11.

[2] Maia, B.M.S., de Assis, M.C.F.R., de Lima, L.M., Rocha, M.B., Calente, H.G., Correa, M.L.A., Camisasca, D.R., Krohling, R.A., 2024. Transformers, convolutional neural networks, and few-shot learning for classification of histopathological images of oral cancer. Expert Systems with Applications 241, 122418.

[3] Liu, Y., Huang, C., 2017. Scene classification via triplet networks. IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing 11, 220–237.

Workflow

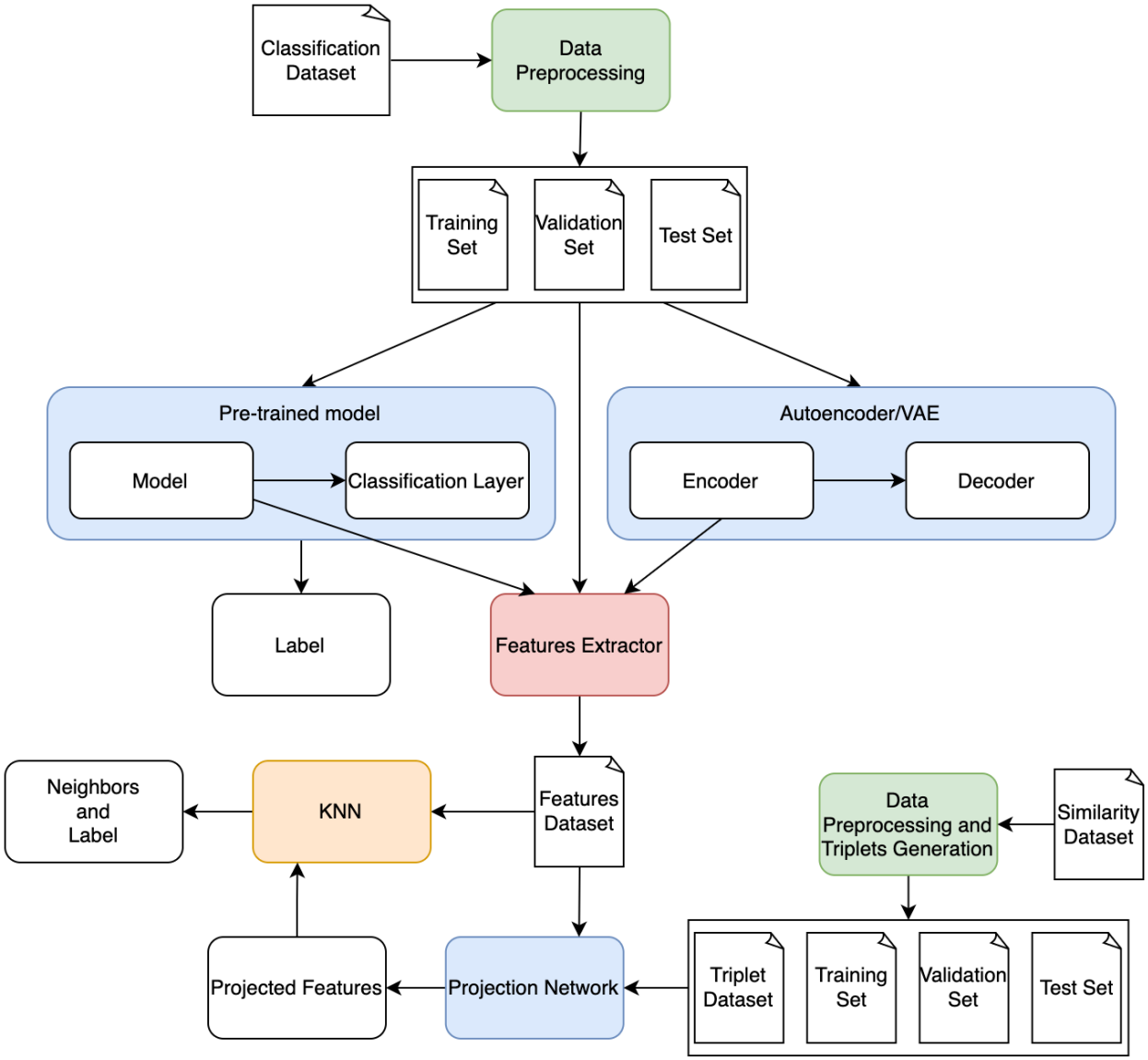
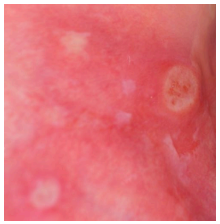
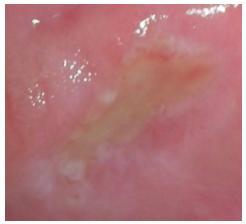
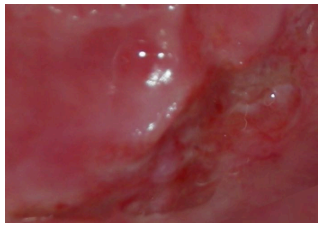

Classification Dataset
Data Preprocessing
Training Set
Validation Set
Test Set
Pre-trained model
Model
Classification Layer
Autoencoder/VAE
Encoder
Decoder
Label
Features Extractor
Neighbors and Label
KNN
Features Dataset
Data Preprocessing and Triplets Generation
Similarity Dataset
Projected Features
Projection Network
Triplet Dataset
Training Set
Validation Set
Test Set

Image Dataset and pre-processing

Pre-processing:

1. check missing images and missing annotations;
2. aggregate image in macro classes:
   - neoplastic
   - aphthous
   - Traumatic

In the end our image dataset is composed of:

Class	Samples
Aphthous	203
Traumatic	206
Neoplastic	179
All	588

3. split dataset

Similarity and Triplet Dataset and preprocessing

Preprocessing:

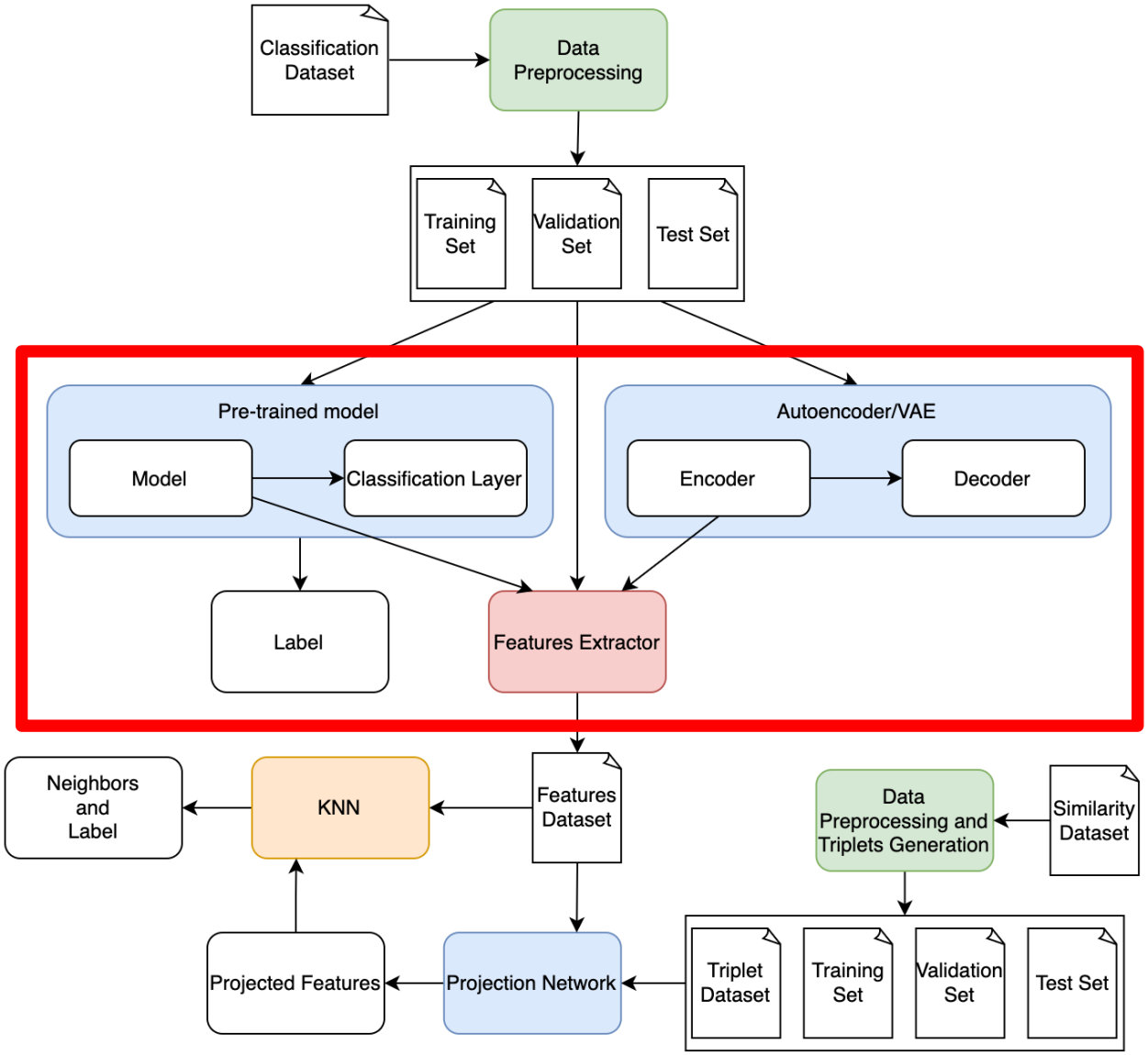
1. Remove: missing or duplicate values, empty rows and rows with missing images
2. create the triplet dataset

We passed from the 208 entries of the similarity dataset to the 29203 entries of the triplet dataset

id_casi	TIPO DI ULCERA	DSCN2465.jpg	DSCN2549.jpg	DSCN2551.jpg	DSCN2702.JPG	DSCN9127.jpg	4-06 033.jpg
id_casi	TIPO DI ULCERA	AFTOSA	AFTOSA	AFTOSA	AFTOSA	AFTOSA	NEOPLASTICA
28-05-2008 013.jpg	AFTOSA	1	3	-1	4	6	18
Immagine 082.jpg	AFTOSA	2	3	-1	4	5	18
DSCN9366.jpg	NEOPLASTICA	-1	14	-1	13	-1	3
08-03-2011_1.jpg	NEOPLASTICA	20	19	18	-1	-1	8

case_id	case_name	type	case_id_pos	case_name_pos	type_pos	rank_pos	case_id_neg	case_name_neg	type_neg	rank_neg
2136	Immagine 082.jpg	AFTOSA	2313	DSCN2549.jpg	AFTOSA	3	2195	DSCN2702.JPG	AFTOSA	4
2136	Immagine 082.jpg	AFTOSA	2192	DSCN2465.jpg	AFTOSA	2	2196	DSCN9127.jpg	AFTOSA	5
2136	Immagine 082.jpg	AFTOSA	2313	DSCN2549.jpg	AFTOSA	3	2216	4-06 033.jpg	NEOPLASTICA	18
2136	Immagine 082.jpg	AFTOSA	2192	DSCN2465.jpg	AFTOSA	2	2313	DSCN2549.jpg	AFTOSA	3
2136	Immagine 082.jpg	AFTOSA	2313	DSCN2549.jpg	AFTOSA	3	2196	DSCN9127.jpg	AFTOSA	5
2136	Immagine 082.jpg	AFTOSA	2192	DSCN2465.jpg	AFTOSA	2	2216	4-06 033.jpg	NEOPLASTICA	18
2136	Immagine 082.jpg	AFTOSA	2196	DSCN9127.jpg	AFTOSA	5	2216	4-06 033.jpg	NEOPLASTICA	18

Workflow


Classification Dataset
Data Preprocessing
Training Set
Validation Set
Test Set
Pre-trained model
Model
Classification Layer
Autoencoder/VAE
Encoder
Decoder
Label
Features Extractor
Neighbors and Label
KNN
Features Dataset
Data Preprocessing and Triplets Generation
Similarity Dataset
Projected Features
Projection Network
Triplet Dataset
Training Set
Validation Set
Test Set

Features Extraction

To extract features we modified the head of classification models:

- linear layer (1): current size -> chosen latent size (in our case 512)
- linear layer (2): chosen latent size -> number of classes

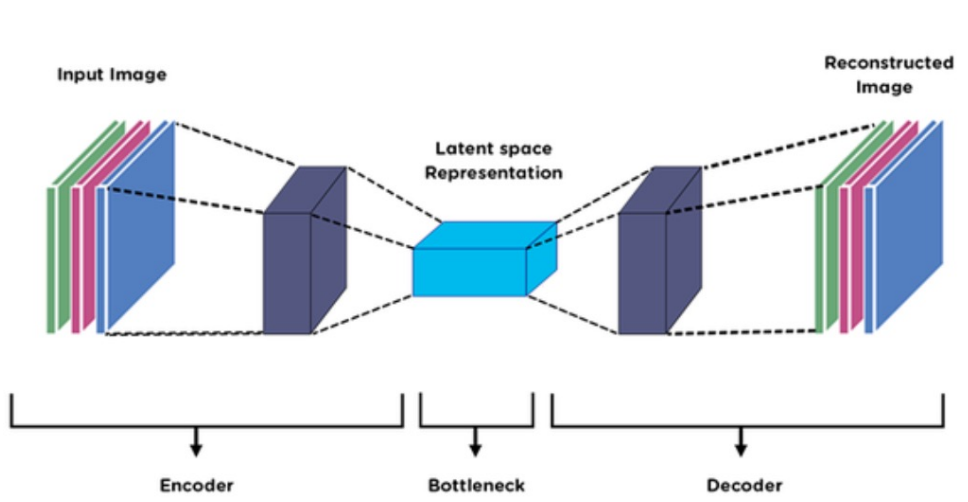
Then we removed the linear layer (2).

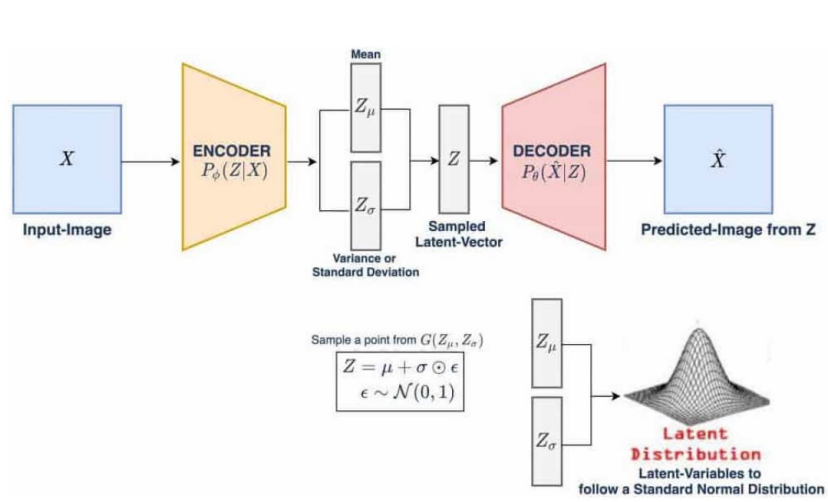
For autoencoder and variational autoencoder, we removed the decoder part.

Classification models

Model	lr	Accuracy	Precision	Recall	F1 score	Epochs
ResNet50	8.0e-05	0.78	0.79	0.78	0.77	86
GoogLeNet	1.0e-05	0.56	0.55	0.56	0.55	320
ViT_B_16	5.0e-06	**0.85**	**0.85**	**0.85**	**0.85**	59
EfficientNet_B1	8.0e-04	0.71	0.73	0.71	0.71	47
DenseNet161	8.0e-06	0.6	0.6	0.6	0.56	257
Swin_T	6.0e-06	0.80	0.80	0.80	0.80	187
VGG13_BN	8.0e-06	0.70	0.70	0.70	0.70	214
RegNet_Y_1_6GF	5.0e-05	0.61	0.71	0.61	0.57	121
MobileNet_V3_Large	8.0e-05	0.65	0.71	0.65	0.61	68
MaxVit_T	1.0e-05	0.75	0.78	0.75	0.76	209
ConvNeXt_Tiny	5.0e-06	0.83	0.84	0.83	0.83	252

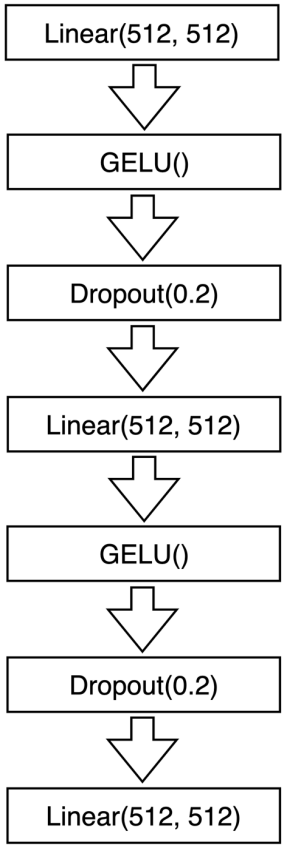
Autoencoders

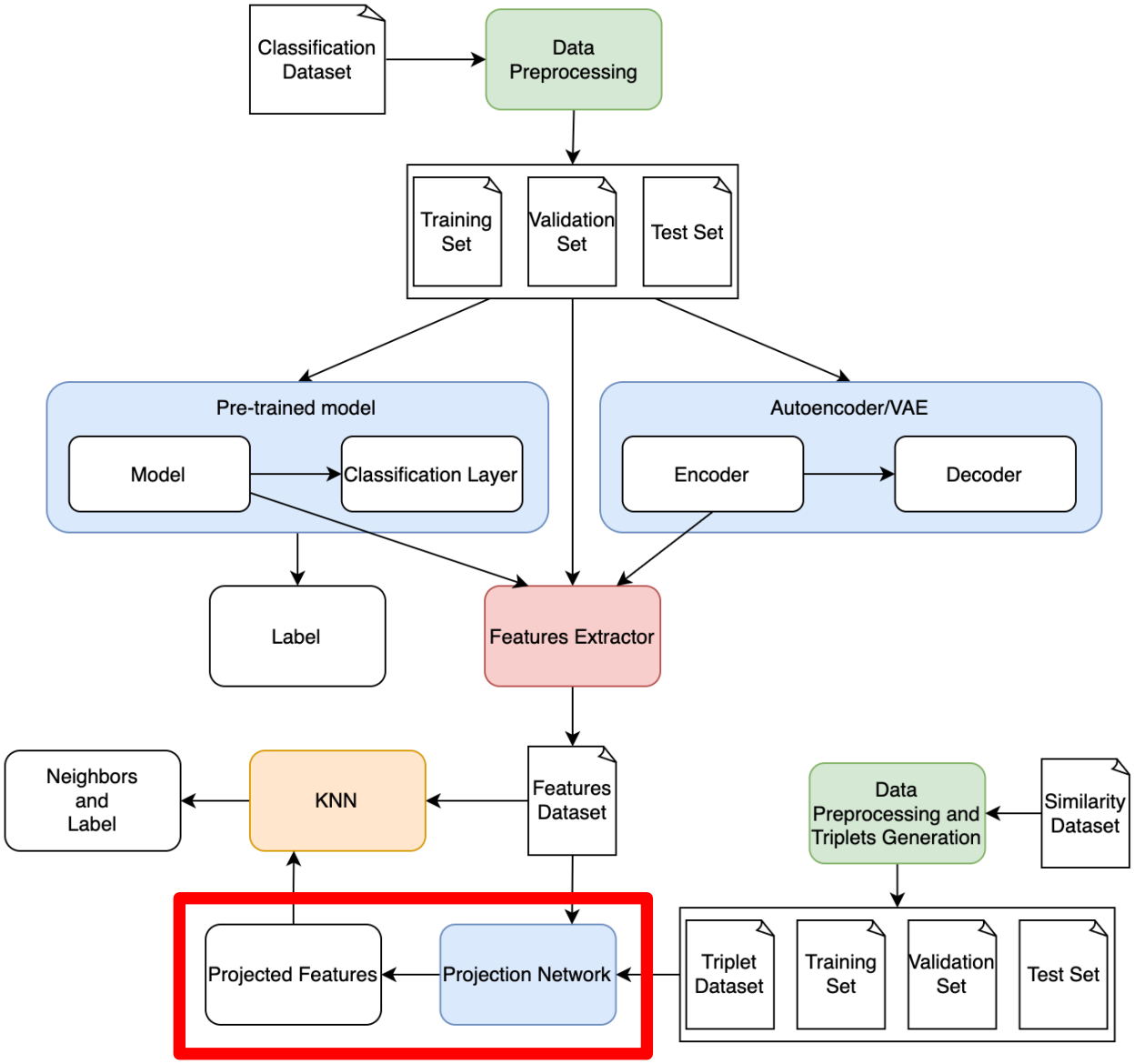




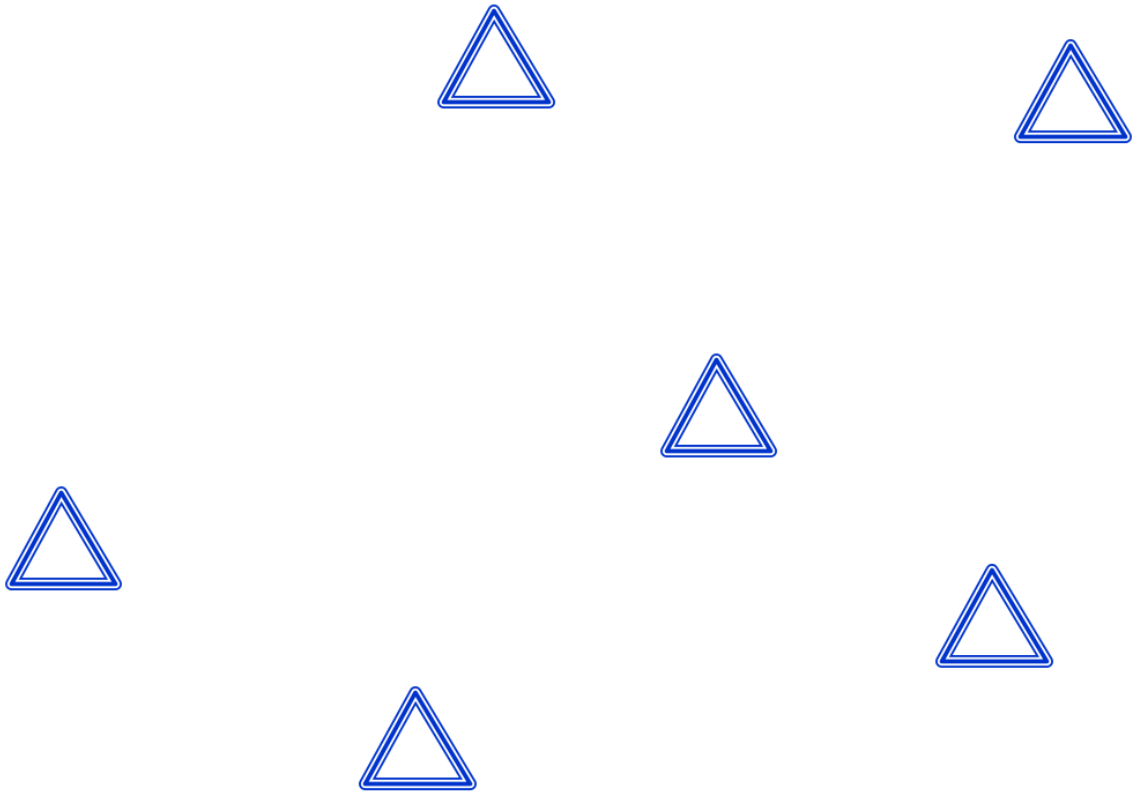
Type	Latent space size	Test loss epoch	Epochs
Image autoencoder	256	0.02899	169
Image autoencoder	512	0.02789	104
VAE	64	1180.35	453
VAE	128	2354.08	686

Workflow

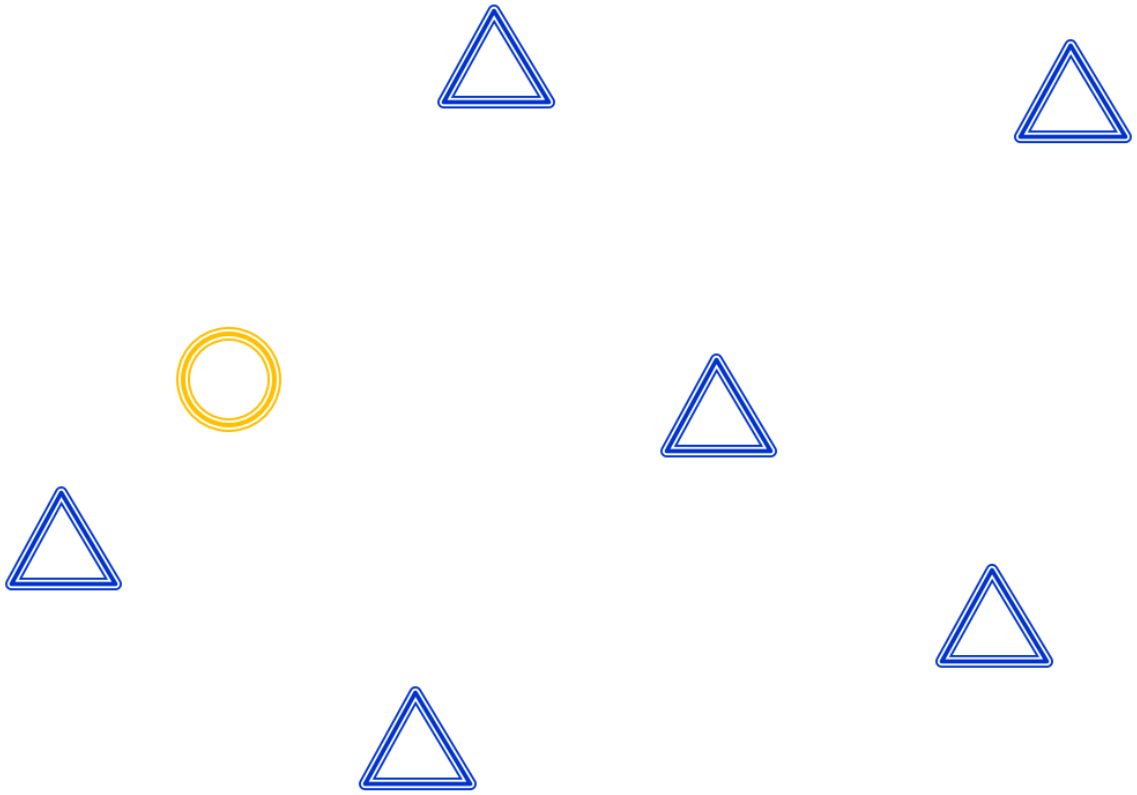

Linear(512, 512)
GELU()
Dropout(0.2)
Linear(512, 512)
GELU()
Dropout(0.2)
Linear(512, 512)


Classification Dataset
Data Preprocessing
Training Set
Validation Set
Test Set
Pre-trained model
Model
Classification Layer
Autoencoder/VAE
Encoder
Decoder
Label
Features Extractor
Neighbors and Label
KNN
Features Dataset
Data Preprocessing and Triplets Generation
Similarity Dataset
Projected Features
Projection Network
Triplet Dataset
Training Set
Validation Set
Test Set

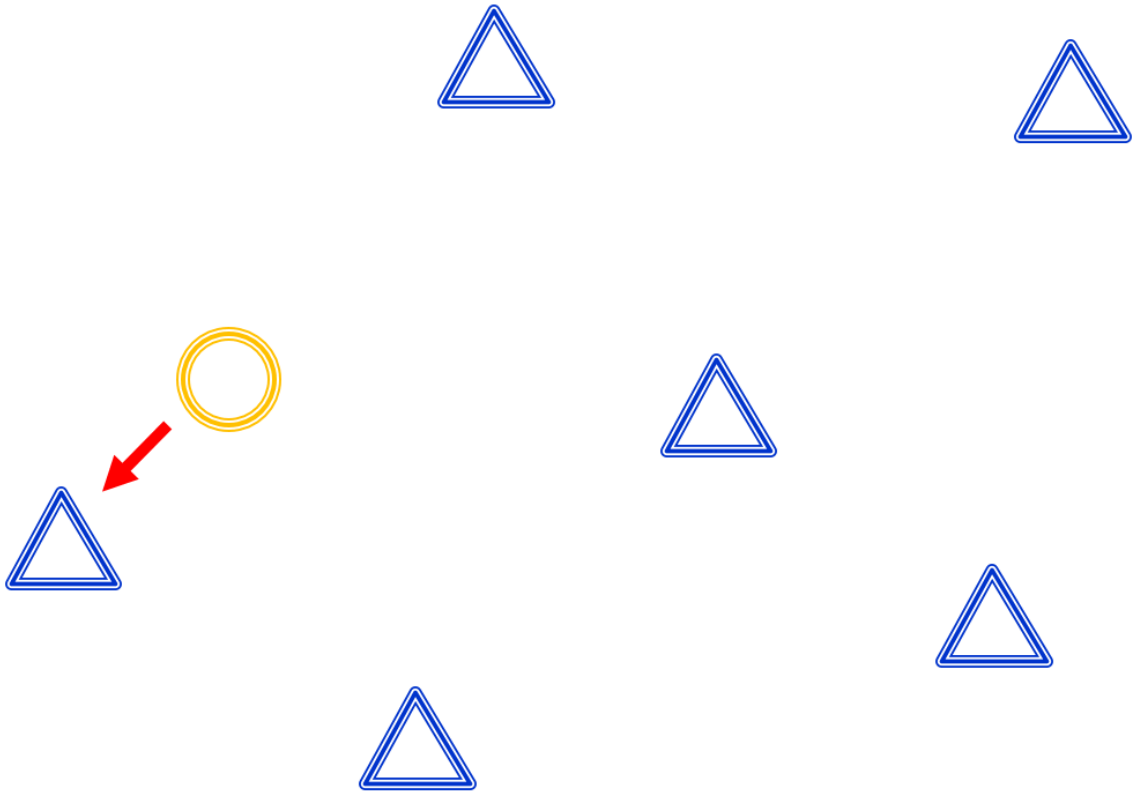
Projection Network



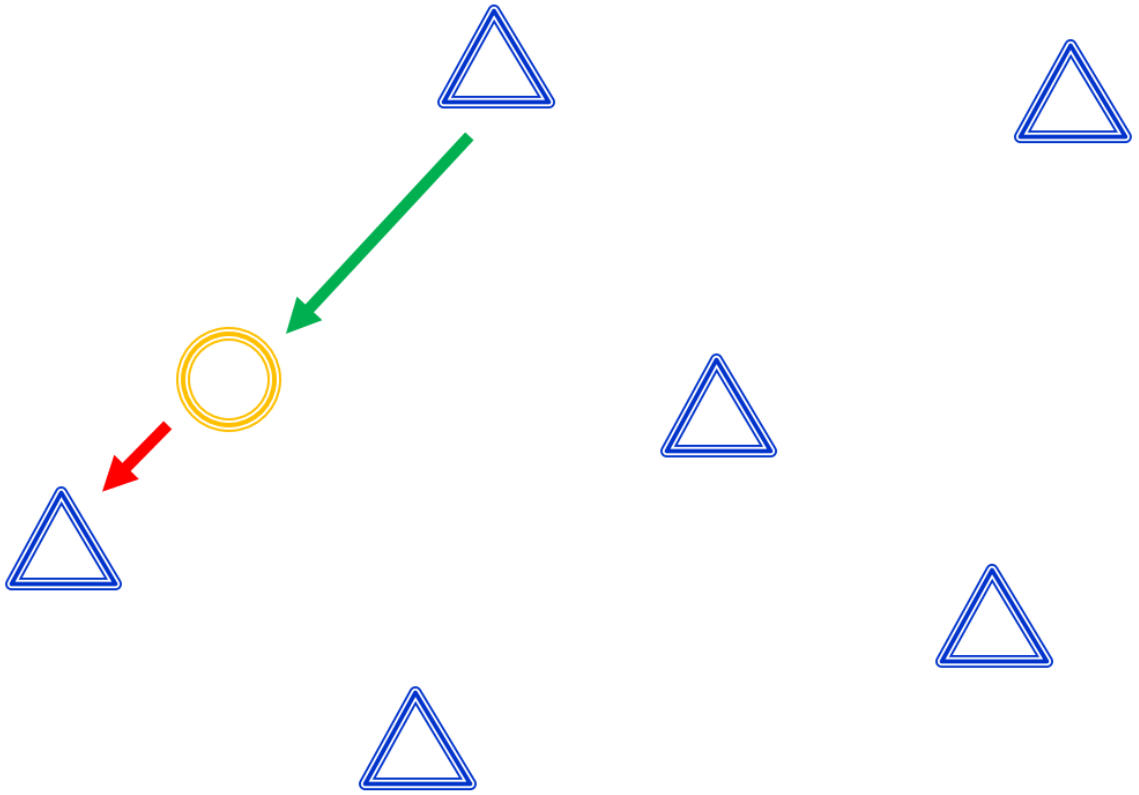
Projection Network



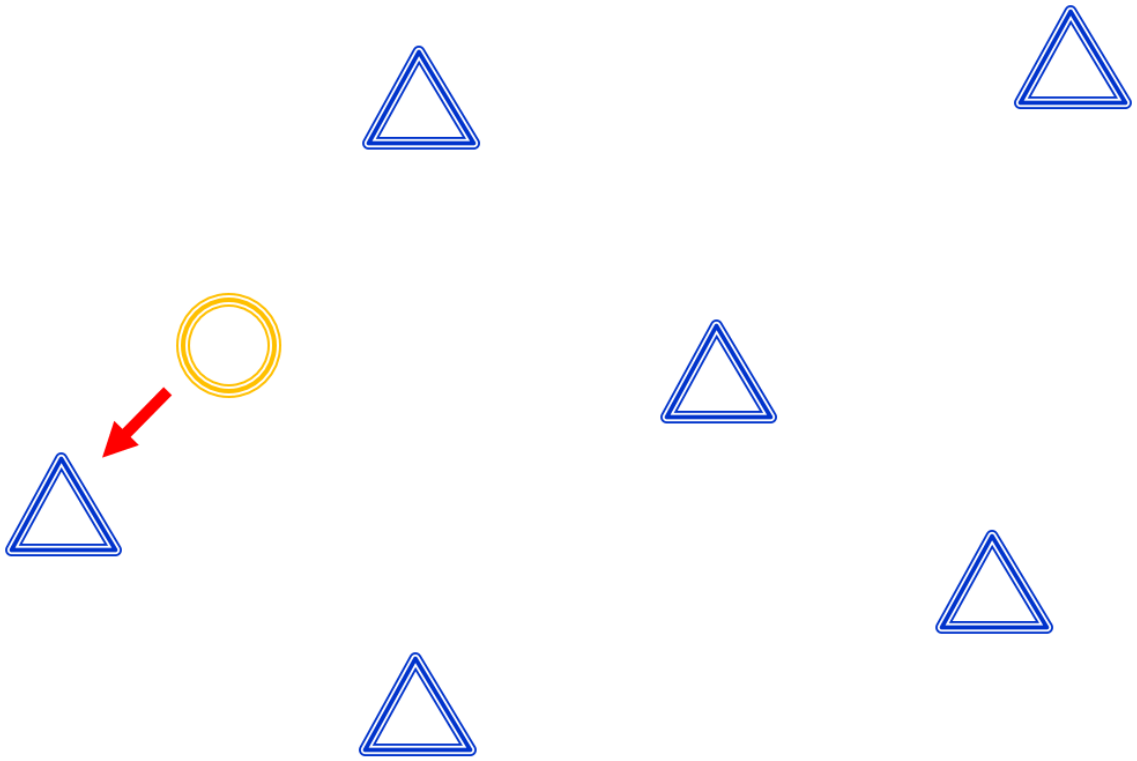
Projection Network

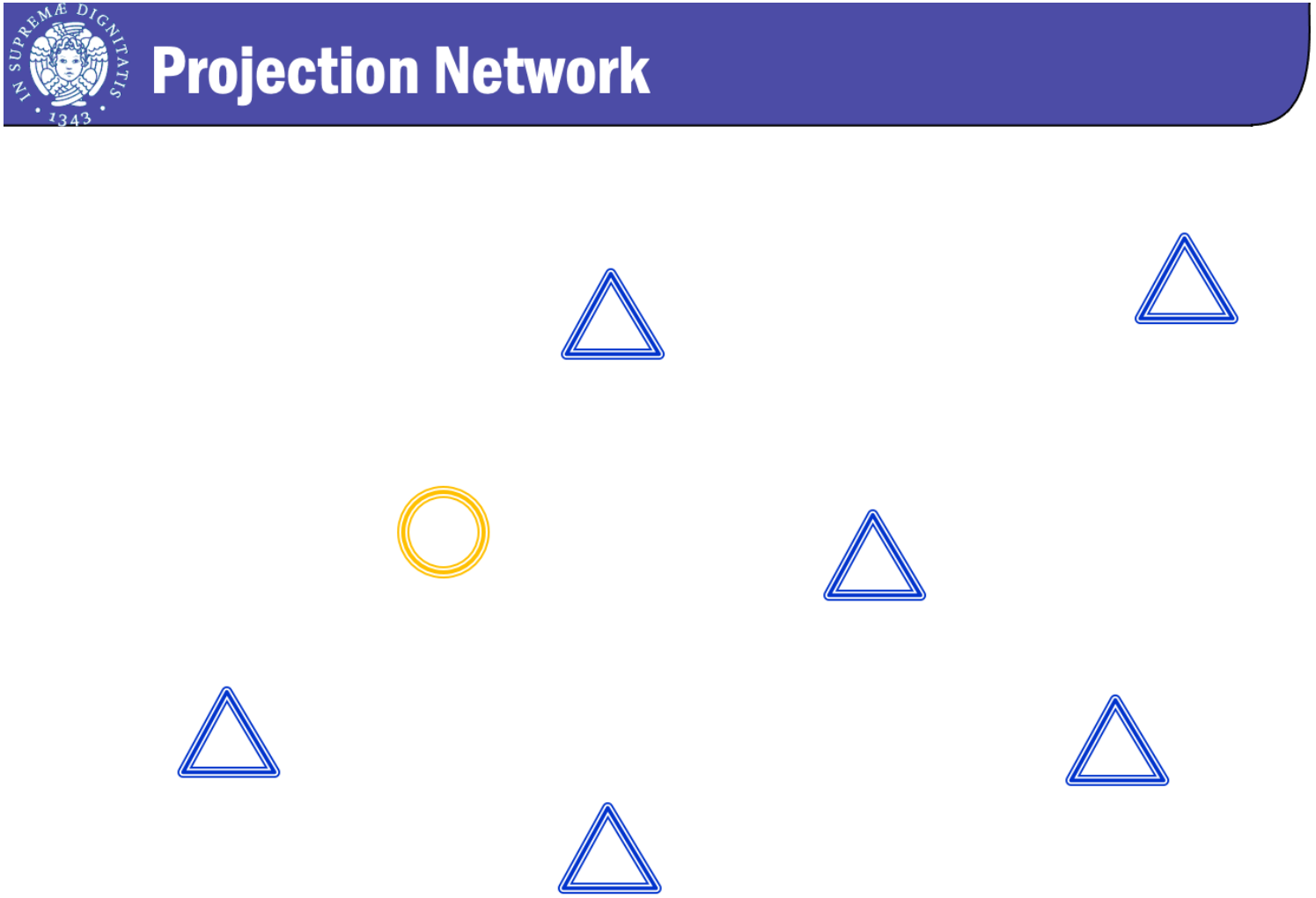


Projection Network

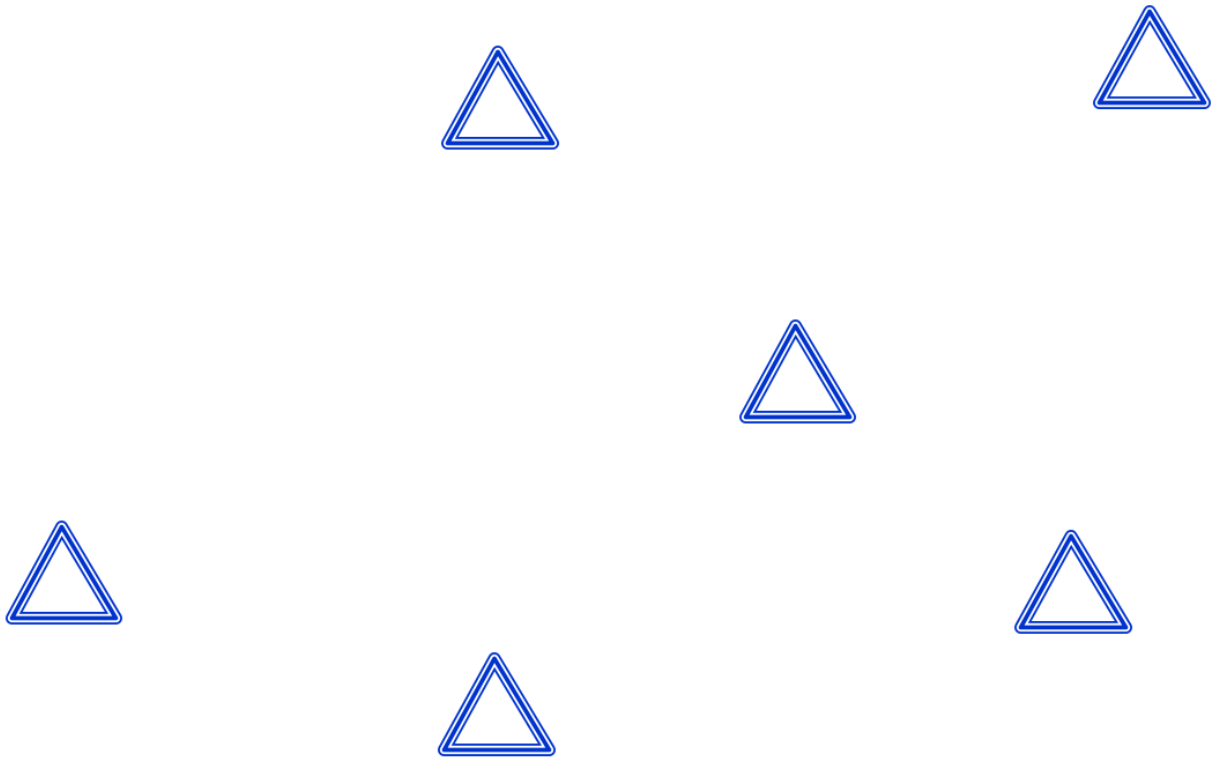


Projection Network

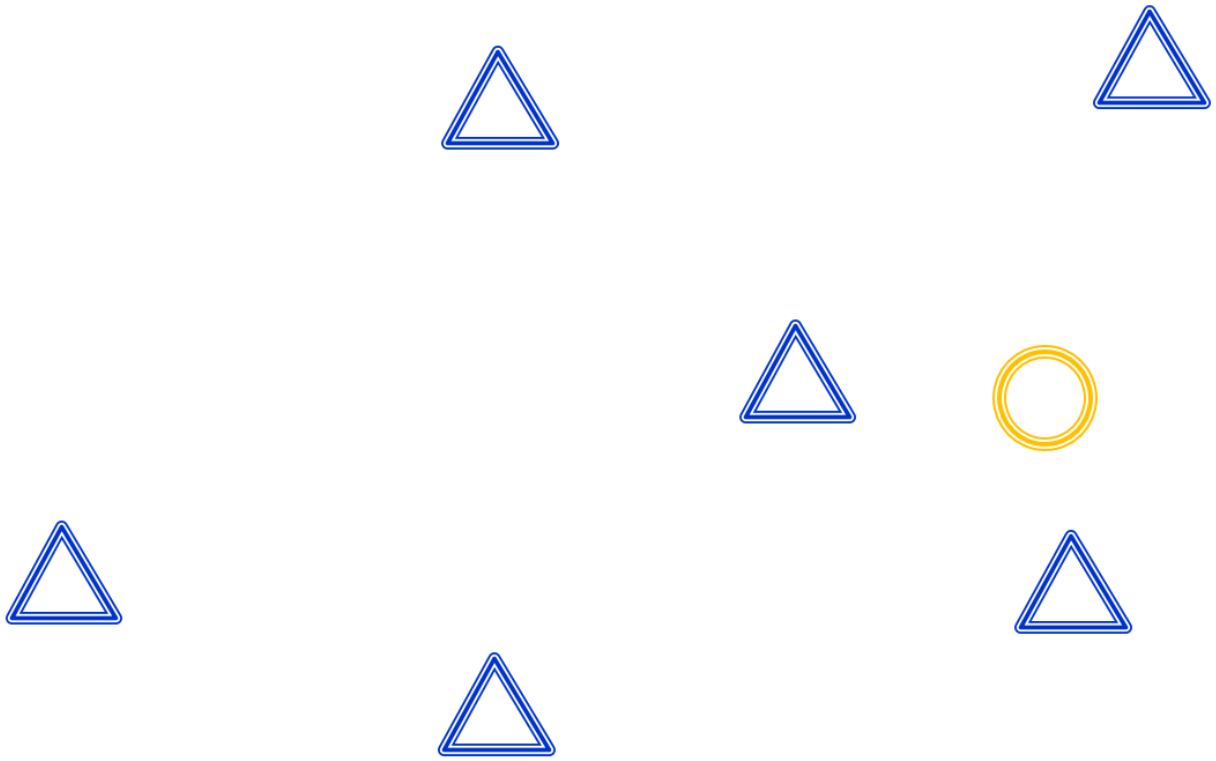



Projection Network

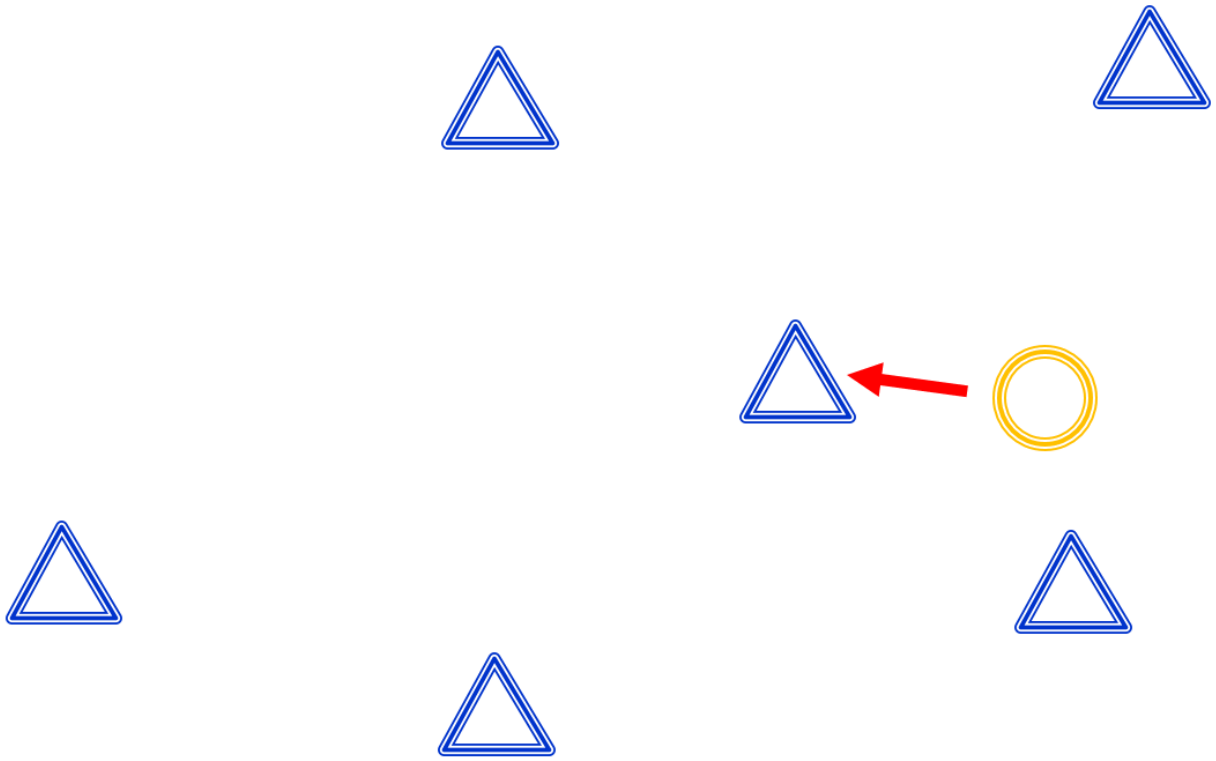
Projection Network



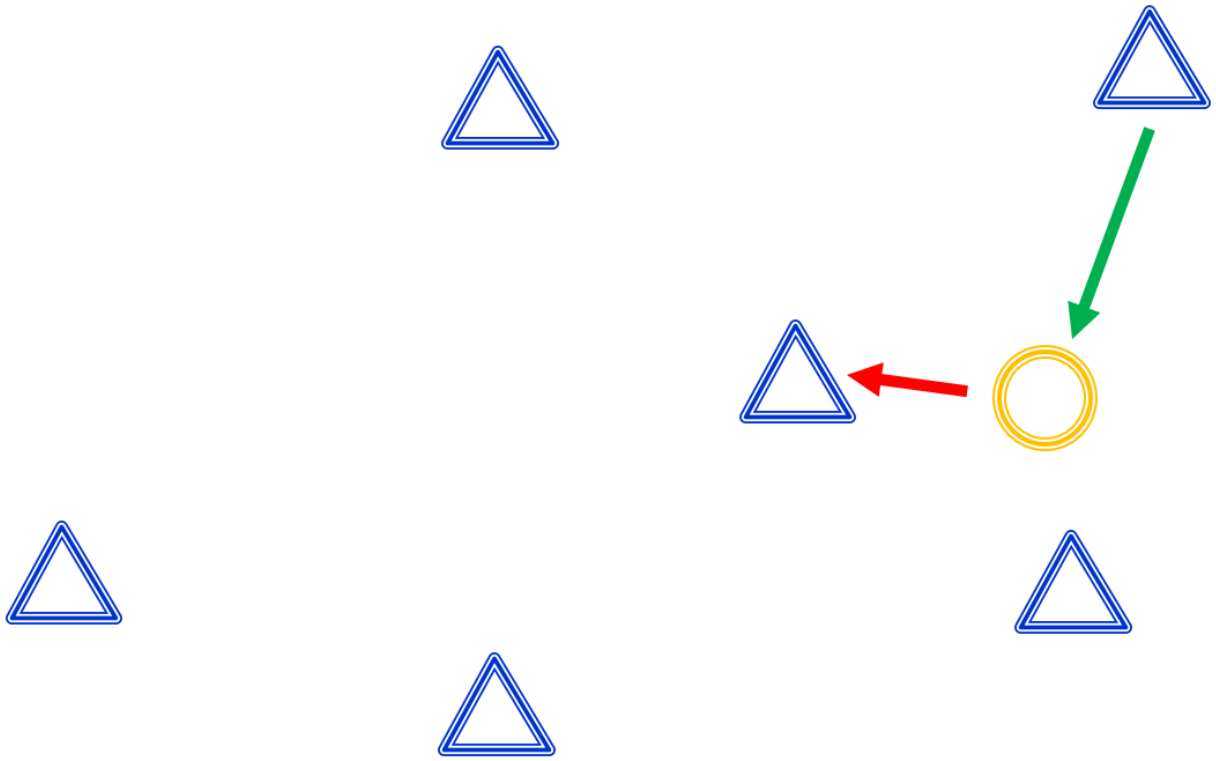
Projection Network



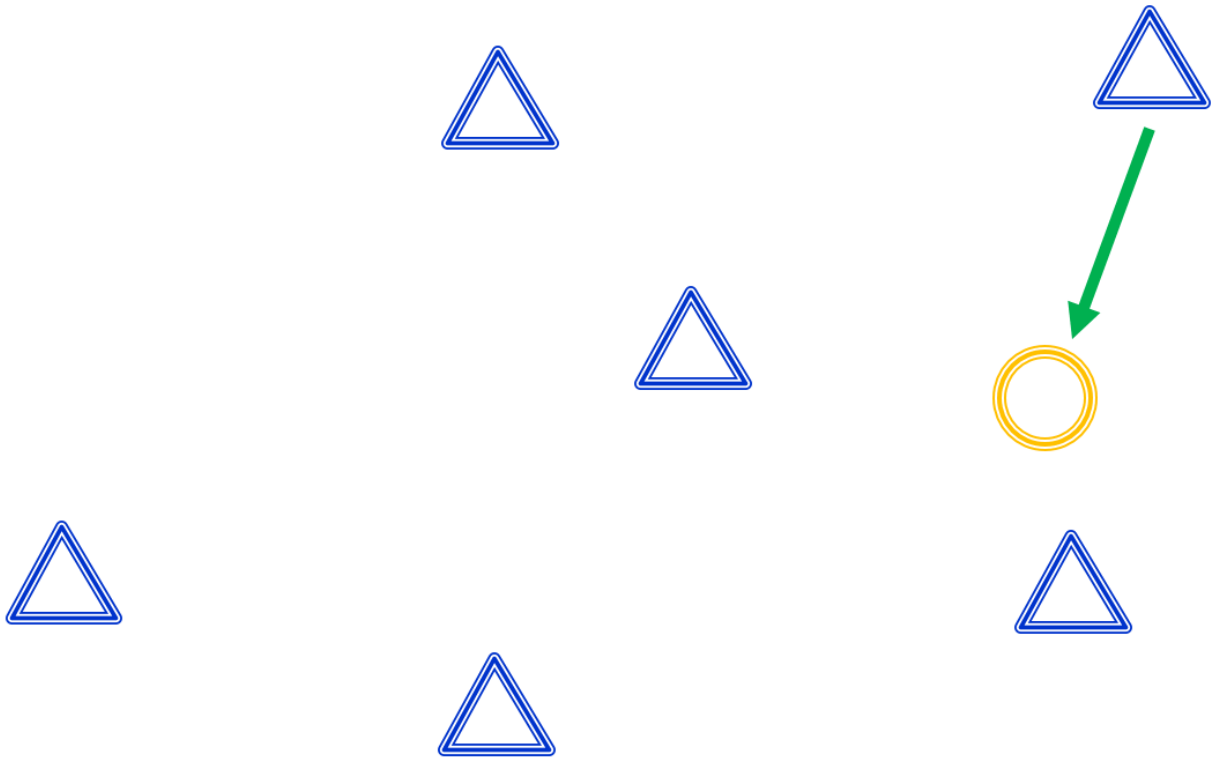
Projection Network



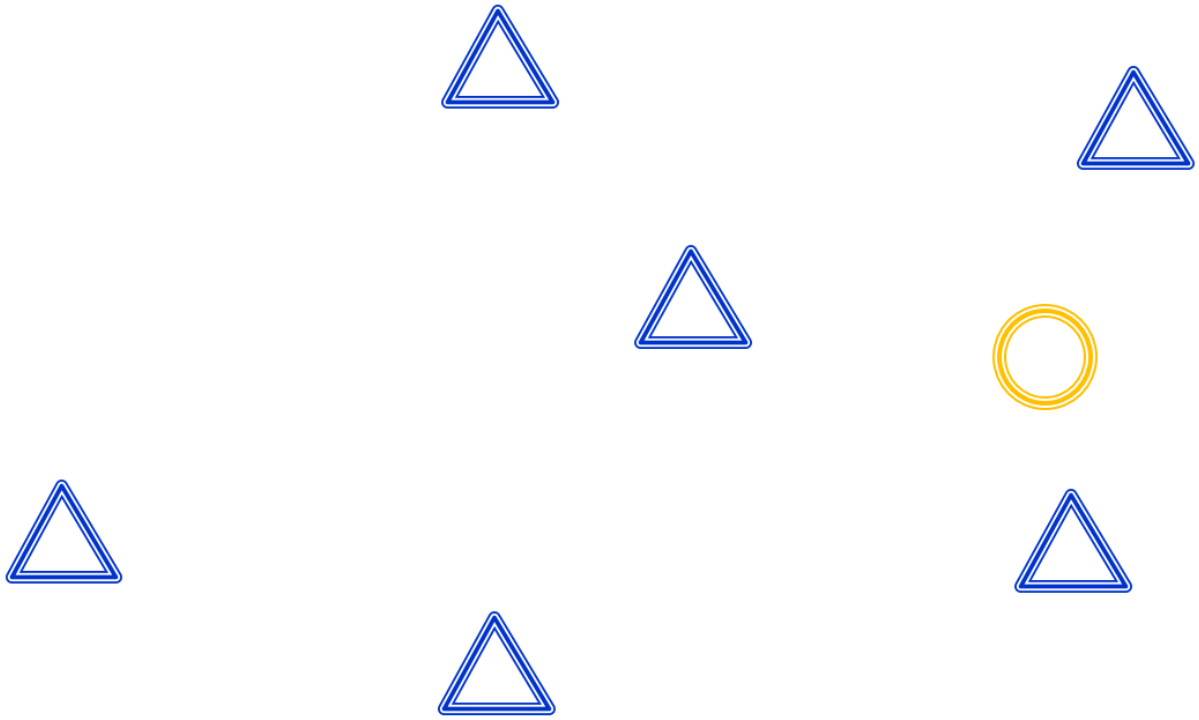
Projection Network



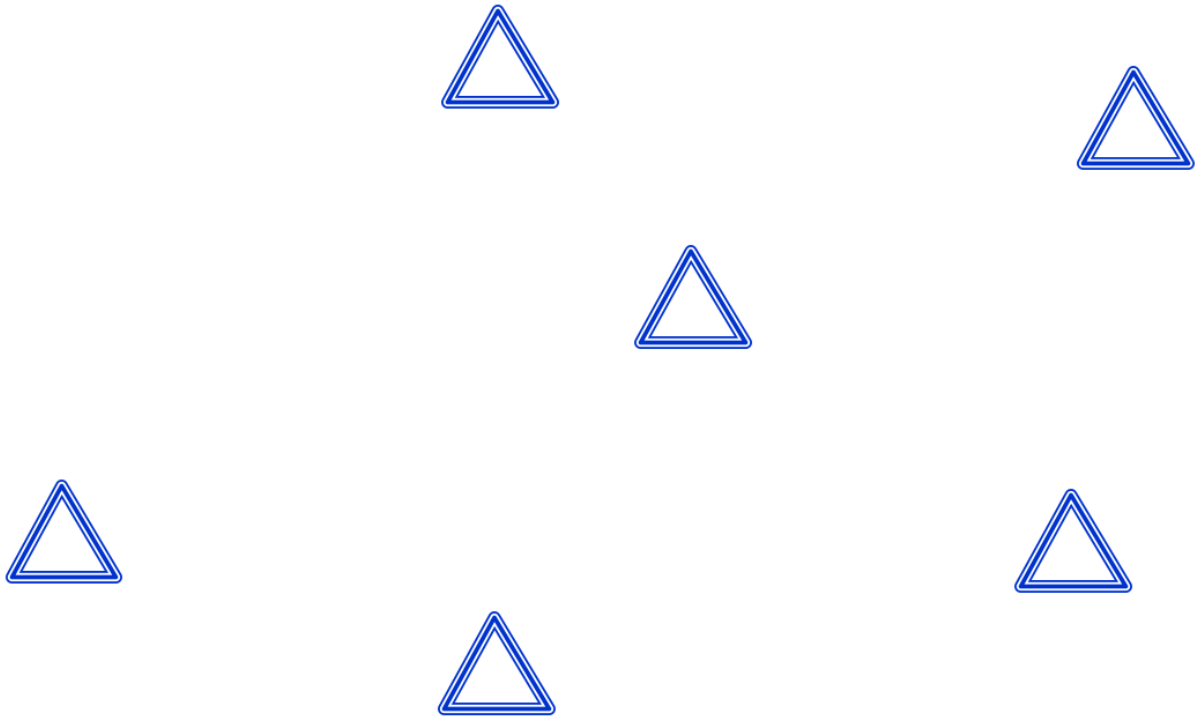
Projection Network



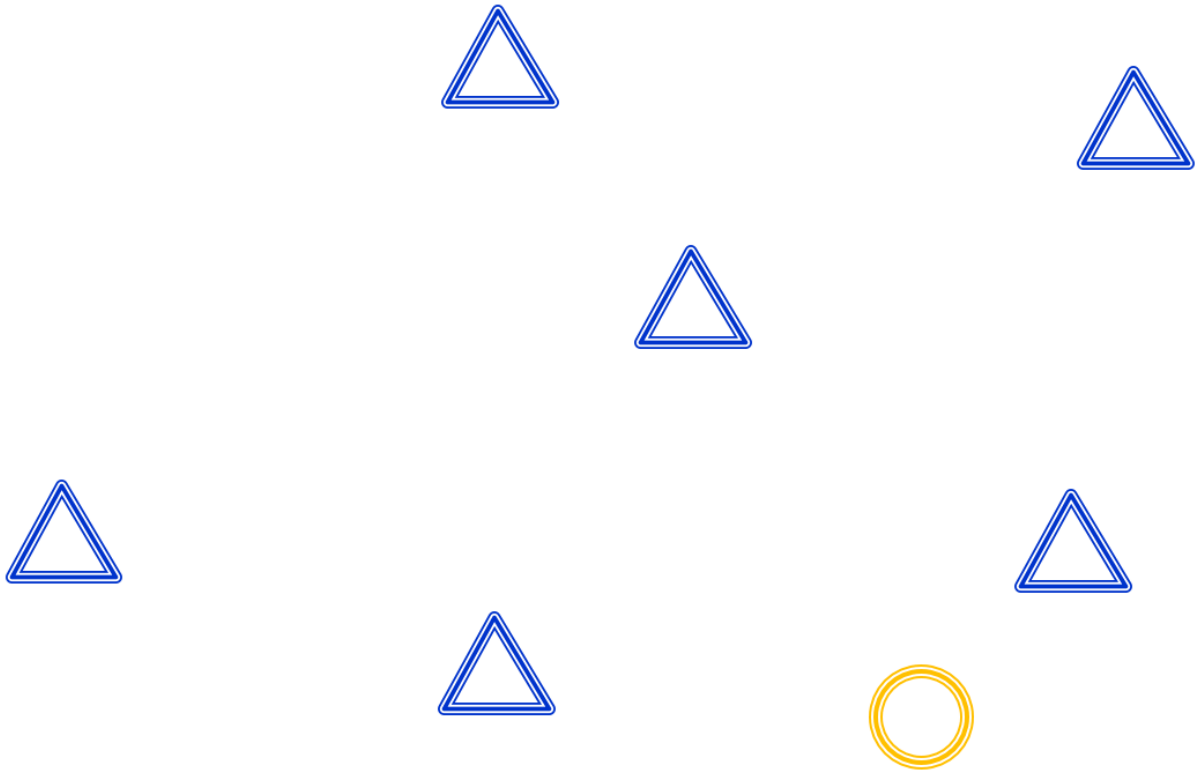
Projection Network



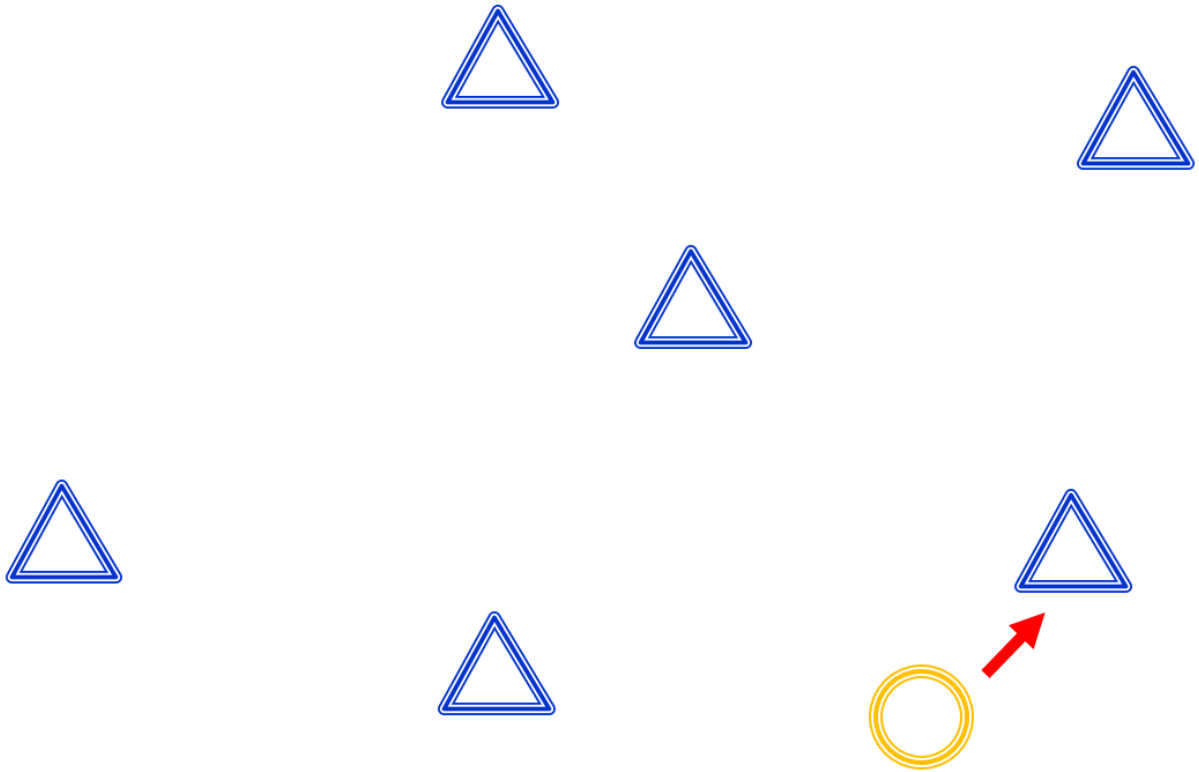
Projection Network



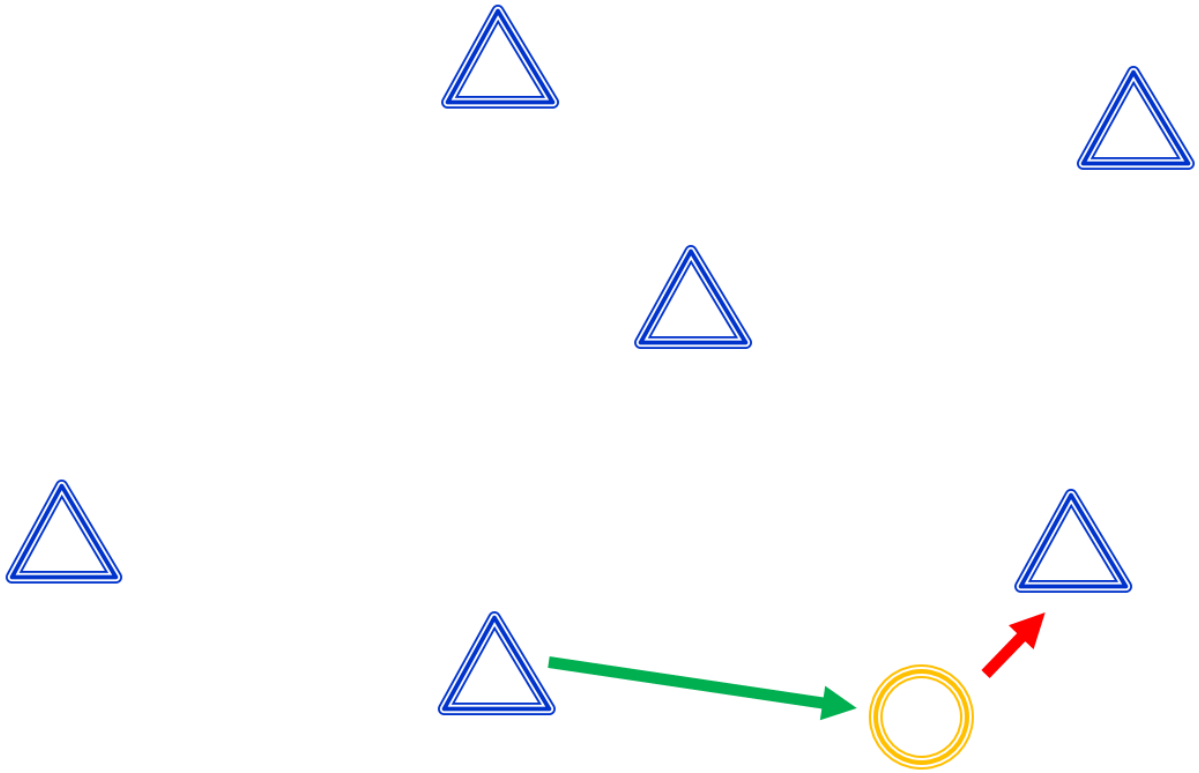
Projection Network



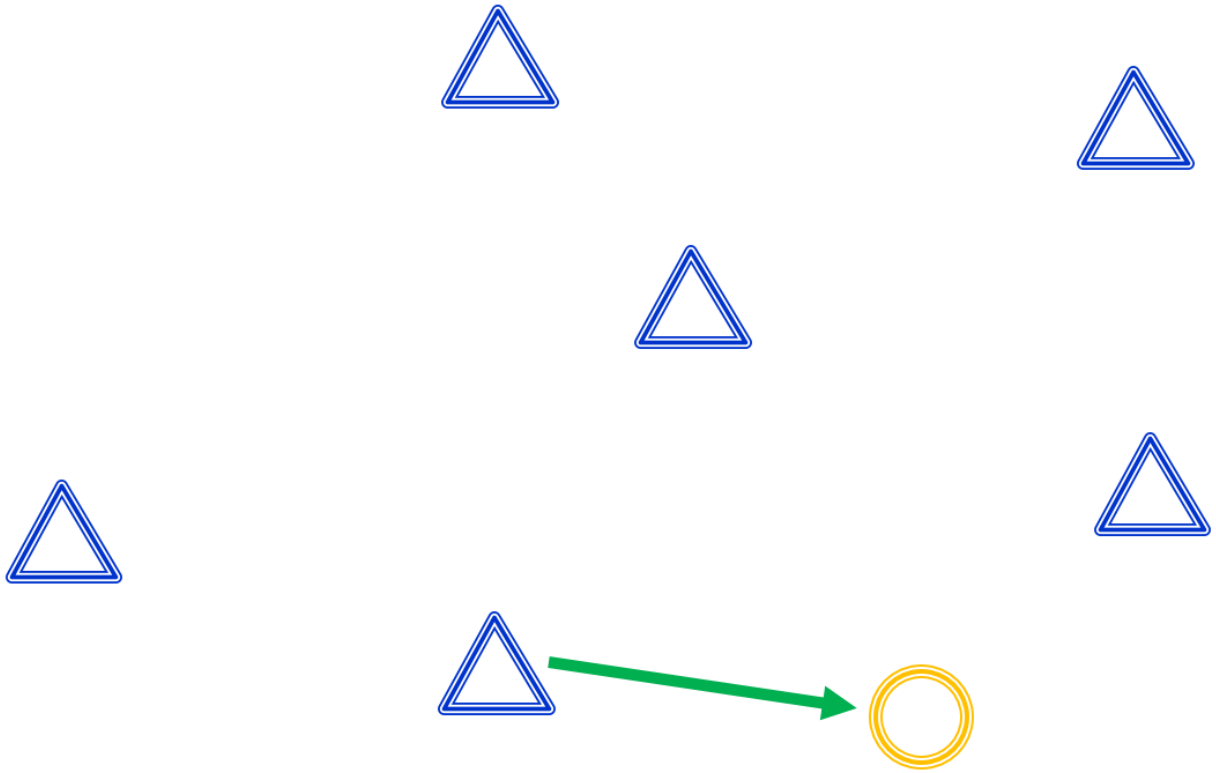
Projection Network



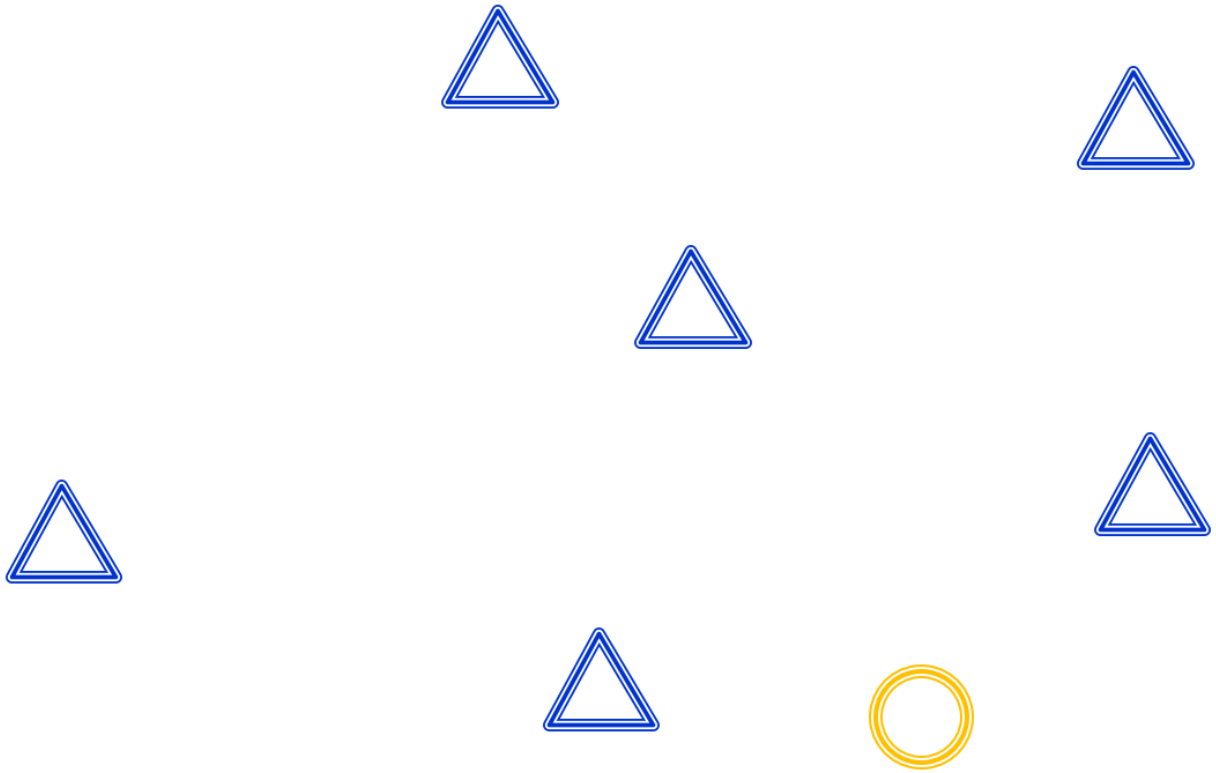
Projection Network



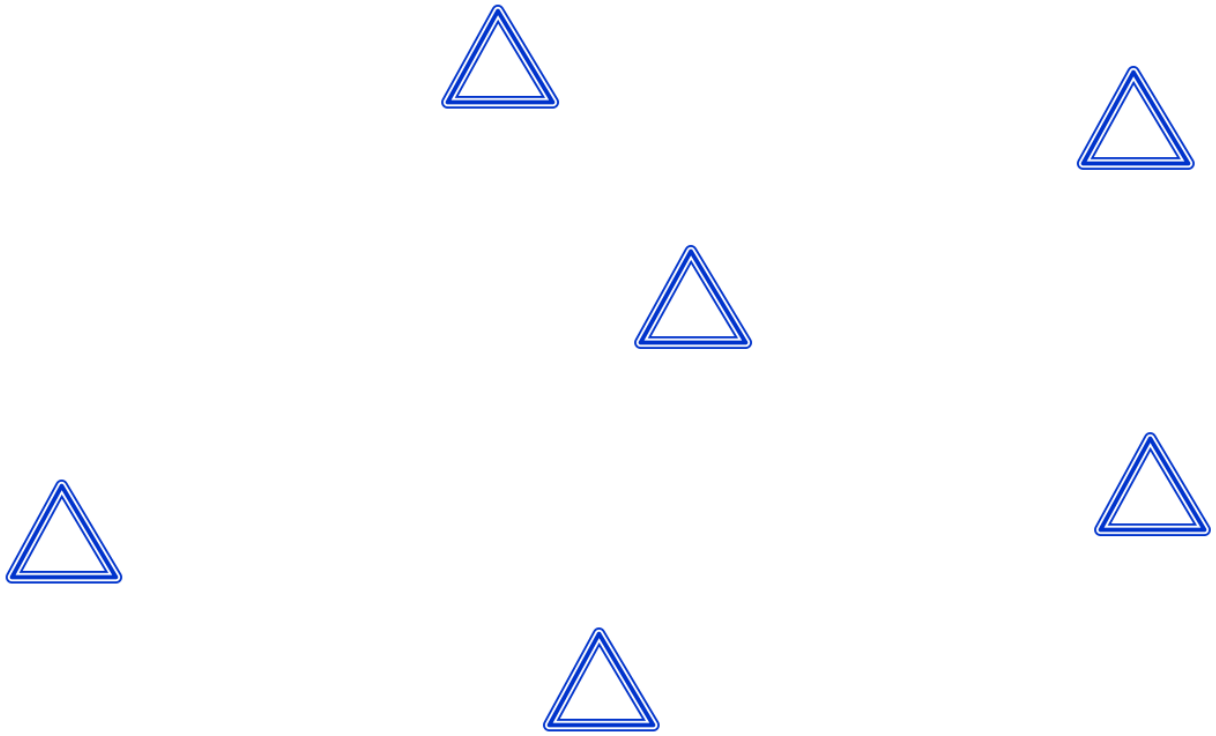
Projection Network



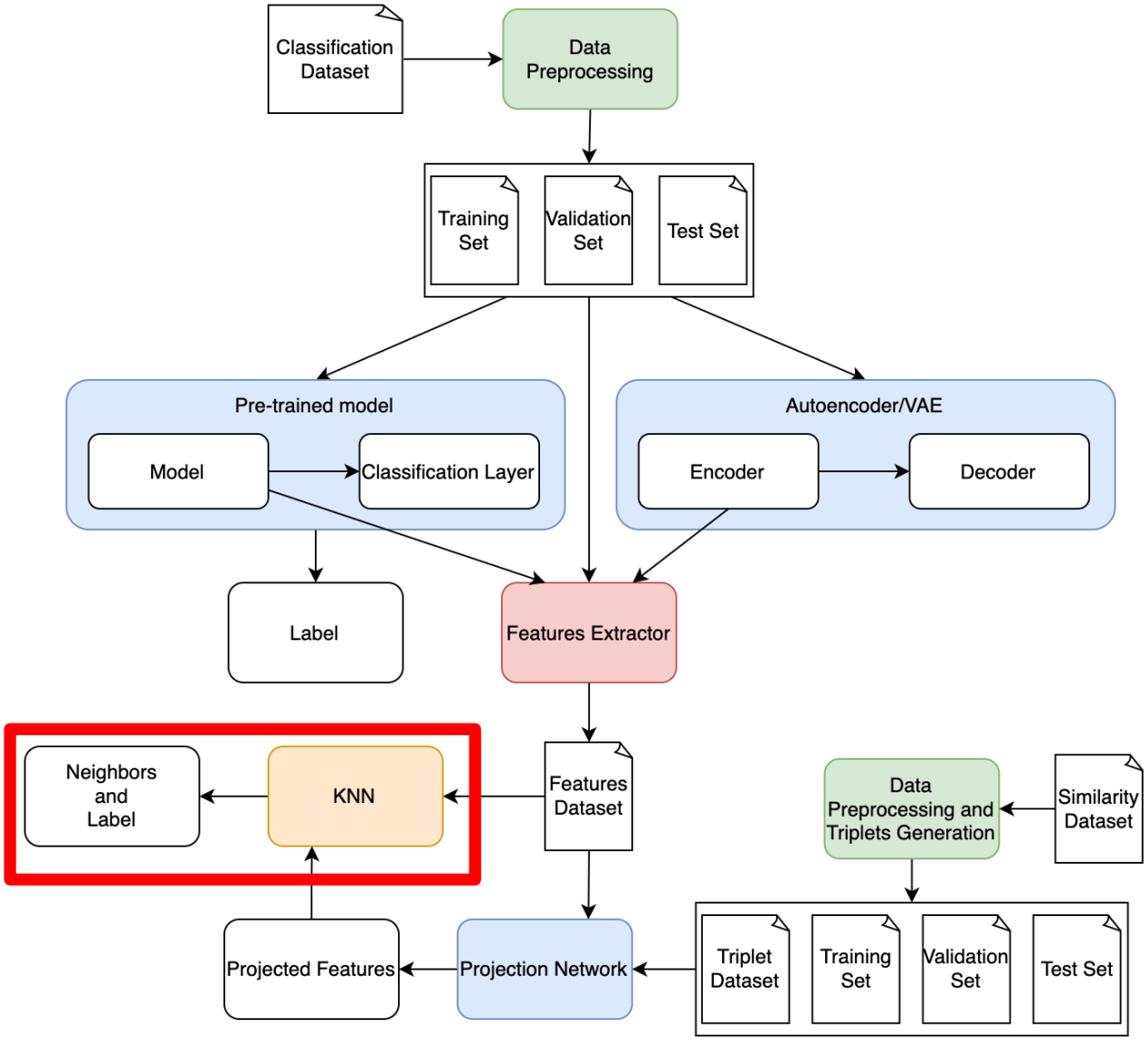
Projection Network



Projection Network



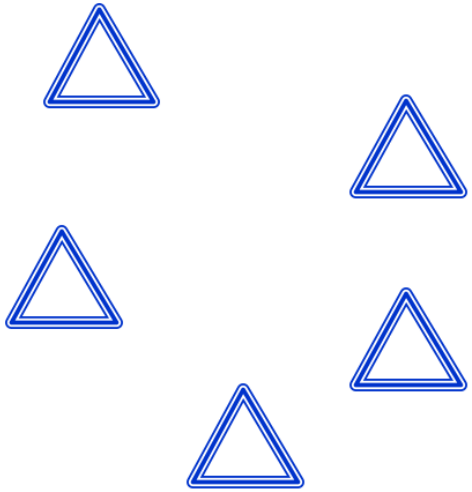
Workflow


Classification Dataset
Data Preprocessing
Training Set
Validation Set
Test Set
Pre-trained model
Model
Classification Layer
Autoencoder/VAE
Encoder
Decoder
Label
Features Extractor
Neighbors and Label
KNN
Features Dataset
Data Preprocessing and Triplets Generation
Similarity Dataset
Projected Features
Projection Network
Triplet Dataset
Training Set
Validation Set
Test Set

Evaluation Process

After the training process:

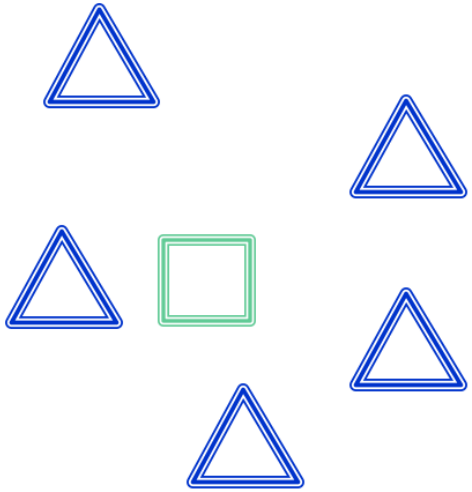
Evaluation Process



After the training process:

1. each image in the test set is projected

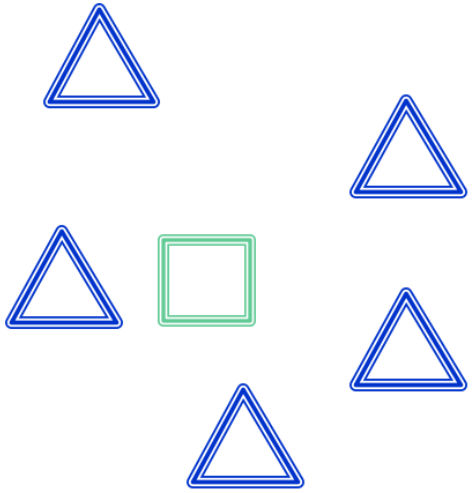
Evaluation Process



After the training process:

1. each image in the test set is projected

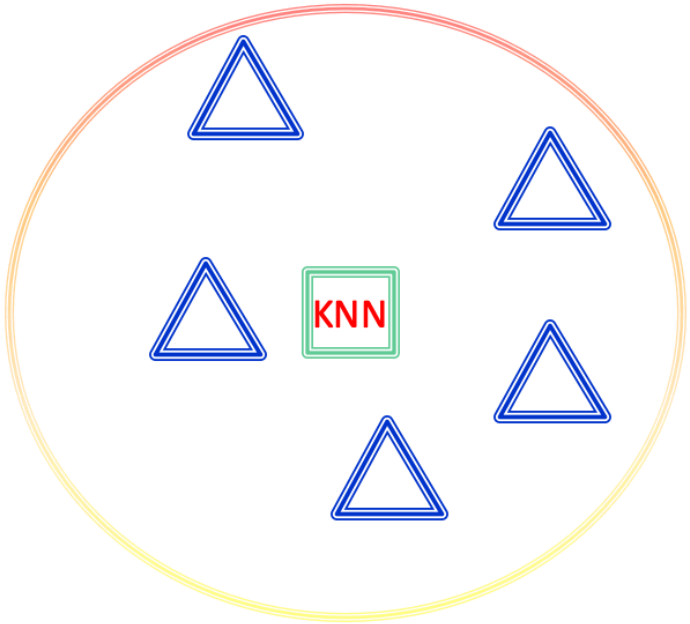
Evaluation Process



After the training process:

1. each image in the test set is projected
2. given an anchor, KNN is computed (k=#reference images)

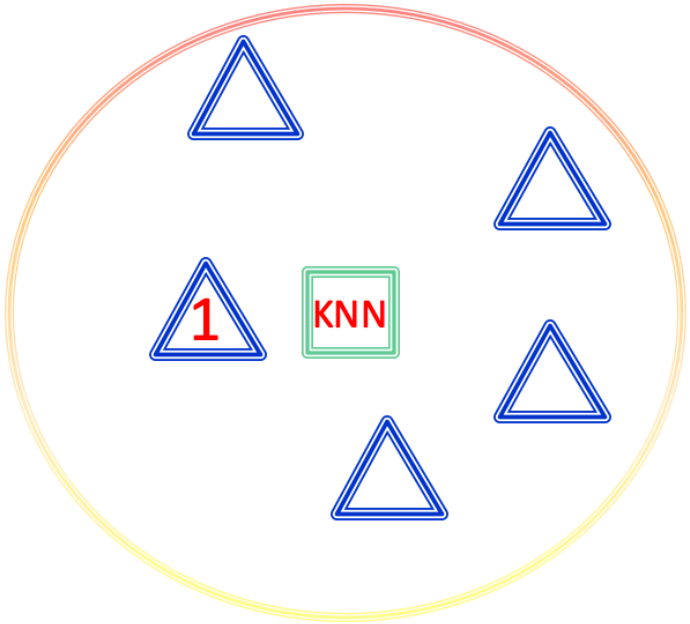
Evaluation Process



After the training process:

1. each image in the test set is projected
2. given an anchor, KNN is computed (k=#reference images)

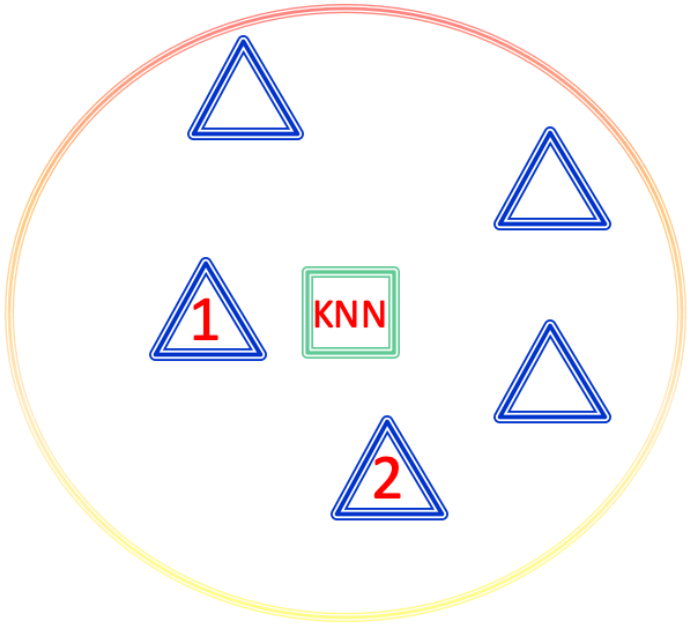
Evaluation Process



After the training process:

1. each image in the test set is projected
2. given an anchor, KNN is computed (k=#reference images)

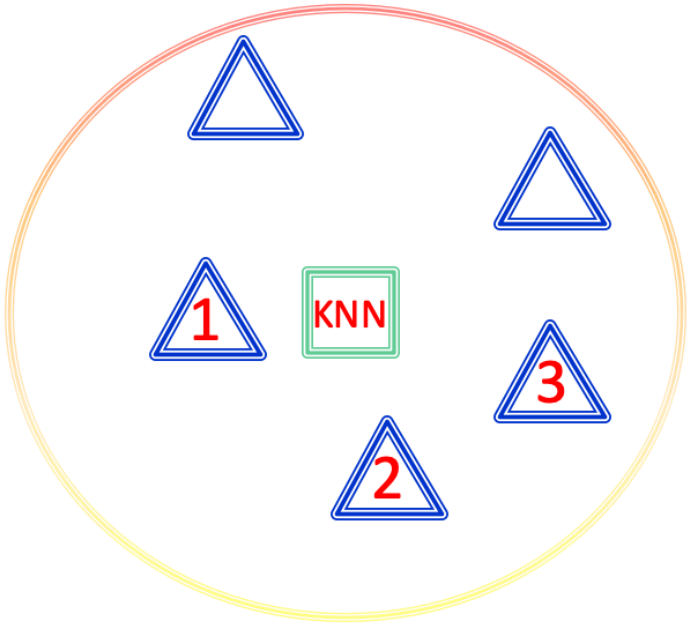
Evaluation Process



After the training process:

1. each image in the test set is projected
2. given an anchor, KNN is computed (k=#reference images)

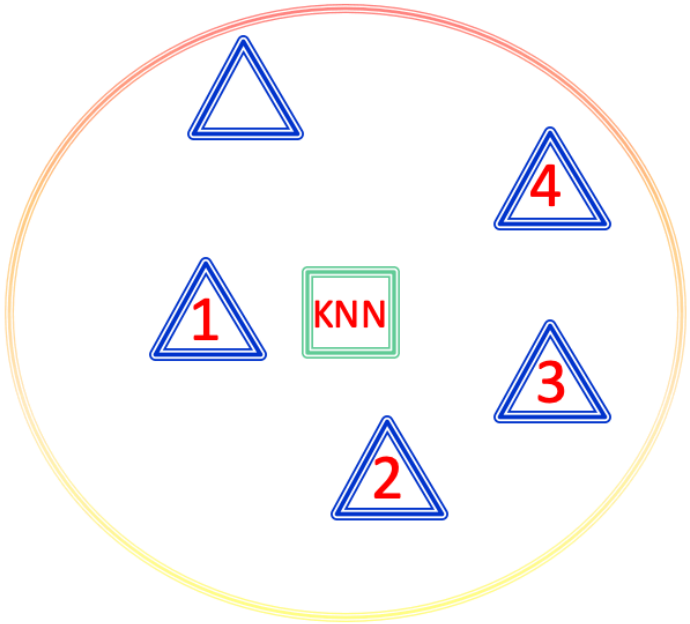
Evaluation Process



After the training process:

1. each image in the test set is projected
2. given an anchor, KNN is computed (k=#reference images)

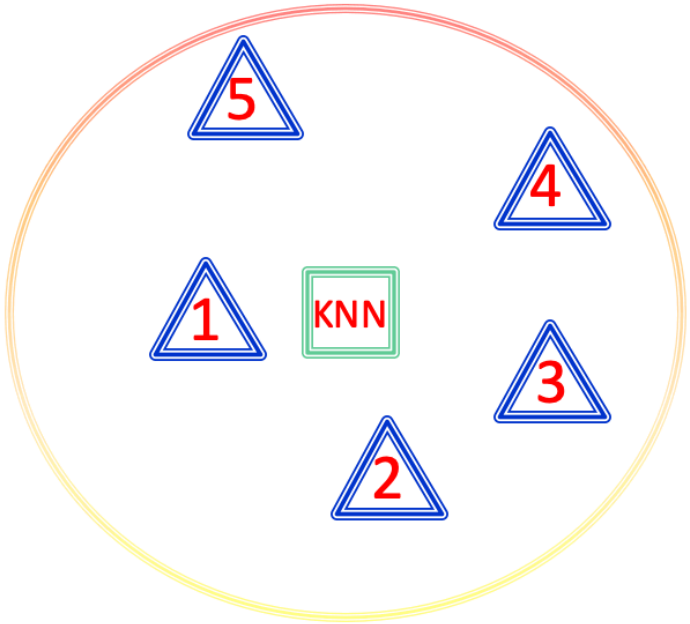
Evaluation Process



After the training process:

1. each image in the test set is projected
2. given an anchor, KNN is computed (k=#reference images)

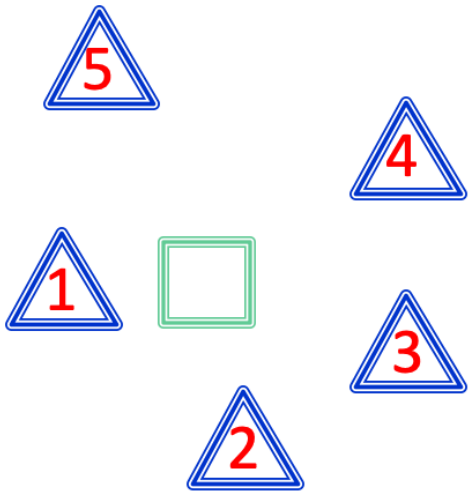
Evaluation Process



After the training process:

1. each image in the test set is projected
2. given an anchor, KNN is computed (k=#reference images)

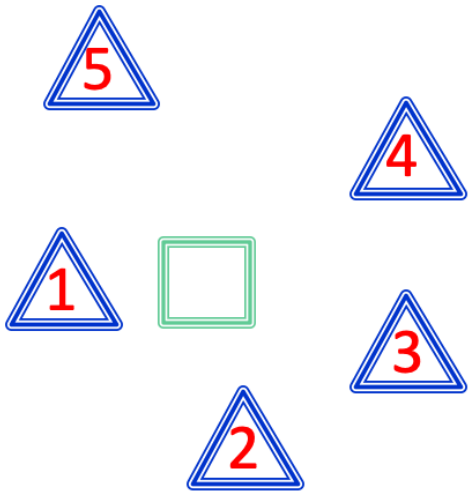
Evaluation Process



After the training process:

1. each image in the test set is projected
2. given an anchor, KNN is computed (k=#reference images)

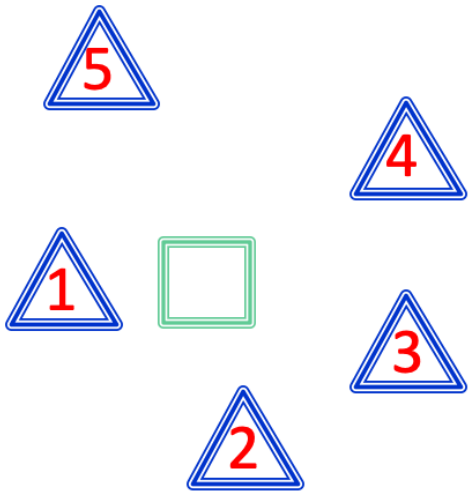
Evaluation Process



After the training process:

1. each image in the test set is projected
2. given an anchor, KNN is computed (k=#reference images)
3. Jaccard distances are computed

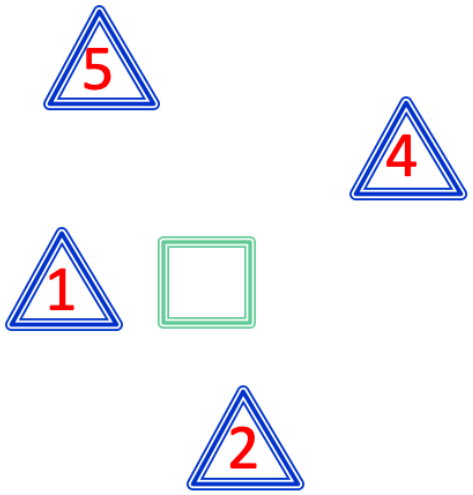
Evaluation Process



After the training process:

1. each image in the test set is projected
2. given an anchor, KNN is computed (k=#reference images)
3. Jaccard distances are computed
4. all reference elements, not in the ground truth, are removed

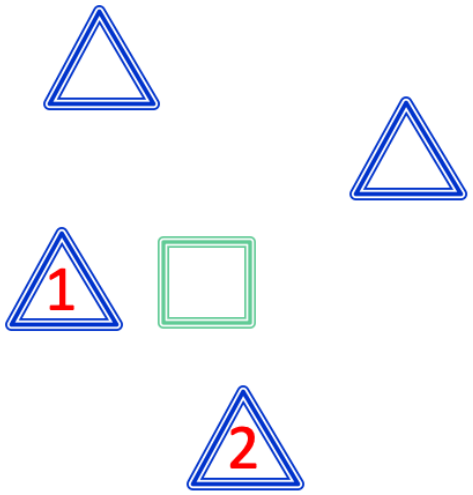
Evaluation Process



After the training process:

1. each image in the test set is projected
2. given an anchor, KNN is computed (k=#reference images)
3. Jaccard distances are computed
4. all reference elements, not in the ground truth, are removed

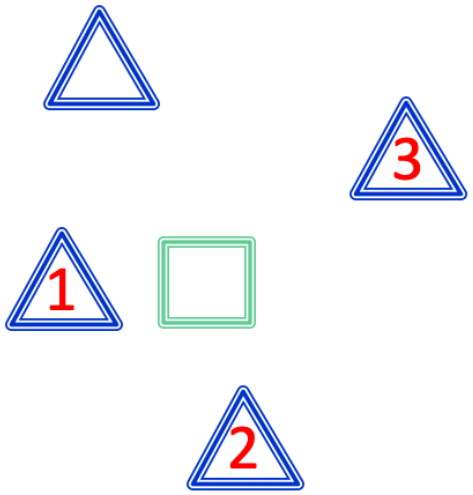
Evaluation Process



After the training process:

1. each image in the test set is projected
2. given an anchor, KNN is computed (k=#reference images)
3. Jaccard distances are computed
4. all reference elements, not in the ground truth, are removed

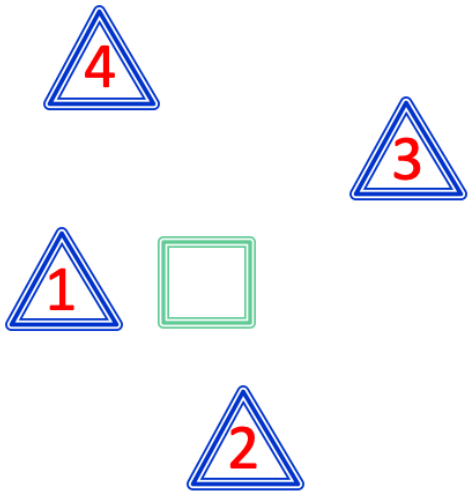
Evaluation Process



After the training process:

1. each image in the test set is projected
2. given an anchor, KNN is computed (k=#reference images)
3. Jaccard distances are computed
4. all reference elements, not in the ground truth, are removed

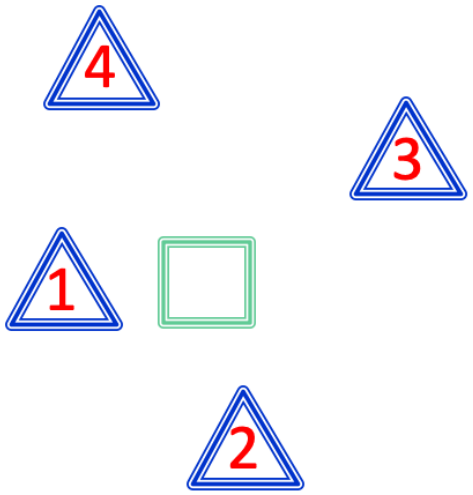
Evaluation Process



After the training process:

1. each image in the test set is projected
2. given an anchor, KNN is computed (k=#reference images)
3. Jaccard distances are computed
4. all reference elements, not in the ground truth, are removed

Evaluation Process



After the training process:

1. each image in the test set is projected
2. given an anchor, KNN is computed (k=#reference images)
3. Jaccard distances are computed
4. all reference elements, not in the ground truth, are removed
5. rest of the metrics are computed on re-ranked values

Pure deep learning results

Model	d_c	d_{sf}	K_τ	d_J	d_J@5	d_J@10	**Accuracy**
ViT_B_16	0.506	0.594	0.440	0.507	0.745	0.592	0.655
ConvNeXt_Tiny	0.453	0.542	0.397	0.439	0.687	0.536	0.851
Swin_T	**0.392**	**0.466**	**0.342**	**0.387**	**0.614**	**0.470**	**0.885**
Autoencoder 256	0.557	0.662	0.492	0.540	0.809	0.643	0.517
Autoencoder 512	0.545	0.649	0.478	0.531	0.791	0.632	0.494
VAE 64	0.573	0.659	0.505	0.578	0.823	0.692	0.448
VAE 128	0.562	0.660	0.493	0.558	0.848	0.685	0.609

Informed deep learning results

Model(Layers)	d_c	d_{sf}	K_τ	d_J	d_J@5	d_J@10	**Accuracy**
ViT_B_16(2)	**0.306**	**0.310**	**0.218**	0.419	**0.492**	0.432	0.851
ViT_B_16(3)	0.341	0.381	0.270	**0.397**	0.593	0.451	0.851
ConvNeXt_Tiny(2)	0.321	0.345	0.242	0.404	0.522	**0.423**	0.897
ConvNeXt_Tiny(3)	0.330	0.358	0.256	0.401	0.589	0.426	0.908
Swin_T(2)	0.324	0.349	0.243	0.406	0.538	0.431	0.908
Swin_T(3)	0.334	0.365	0.253	0.410	0.575	0.431	**0.931**
Autoencoder256(2)	0.439	0.484	0.356	0.506	0.660	0.552	0.517
Autoencoder256(3)	0.445	0.489	0.363	0.511	0.683	0.545	0.552
Autoencoder512(2)	0.377	0.406	0.291	0.461	0.568	0.484	0.736
Autoencoder512(3)	0.372	0.403	0.289	0.452	0.568	0.475	0.736
VAE64(2)	0.519	0.580	0.424	0.585	0.784	0.664	0.230
VAE64(3)	0.538	0.618	0.456	0.569	0.784	0.659	0.253
VAE128(2)	0.448	0.481	0.354	0.540	0.726	0.600	0.356
VAE128(3)	0.478	0.527	0.381	0.559	0.756	0.628	0.322

Conclusion

Conclusions:

- Classification models perform better than autoencoders in the projection
- Projection with a lower number of layers obtained better results networks
- For some models this approach is not able to maintain/increase the accuracy score

Other works:

- Test different classification models
- Test greater latent dimensions