Исследование эффективности различных методов обучения моделей классификации изображений в условиях малой обучающей выборки (few-shot learning) на примере классификации болезней растений

Александр Ужинский
ЛИТМ, ОИЯИ
auzhinskiy@jinr.ru

Revolution

Industry 1.0

Mechanization, steam power, weaving loom

1784

Industry 2.0

Mass production, assembly line, electrical energy

1870

Industry 3.0

Automation, computers and electronics

1969

Industry 4.0

Cyber physical systems, internet of things (IoT), networks

Today

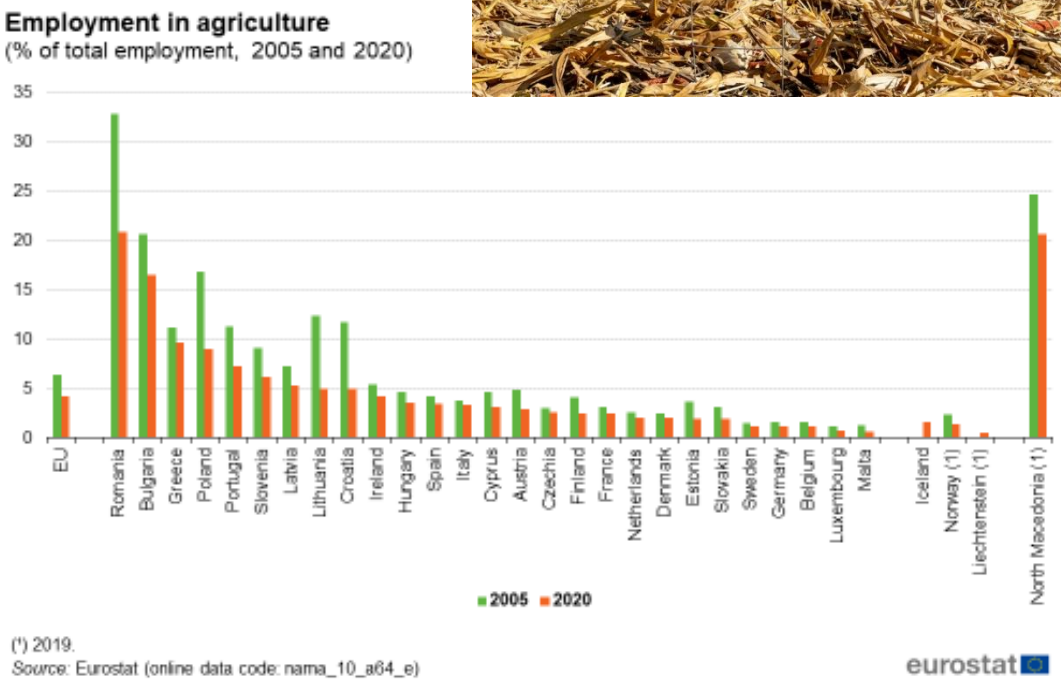
Advanced technologies in agriculture (Agriculture 4.0)

- IoT, sensors,
- remote sensing,
- big-data analysis,
- robots,
- drones,
- digitalization,
- artificial intelligence,
- etc.

There are also many interesting projects in chemistry-, biology-, genetic- and other areas

Animal husbandry is very interesting area with great impact of advanced technologies, but it is out of scope of the report!

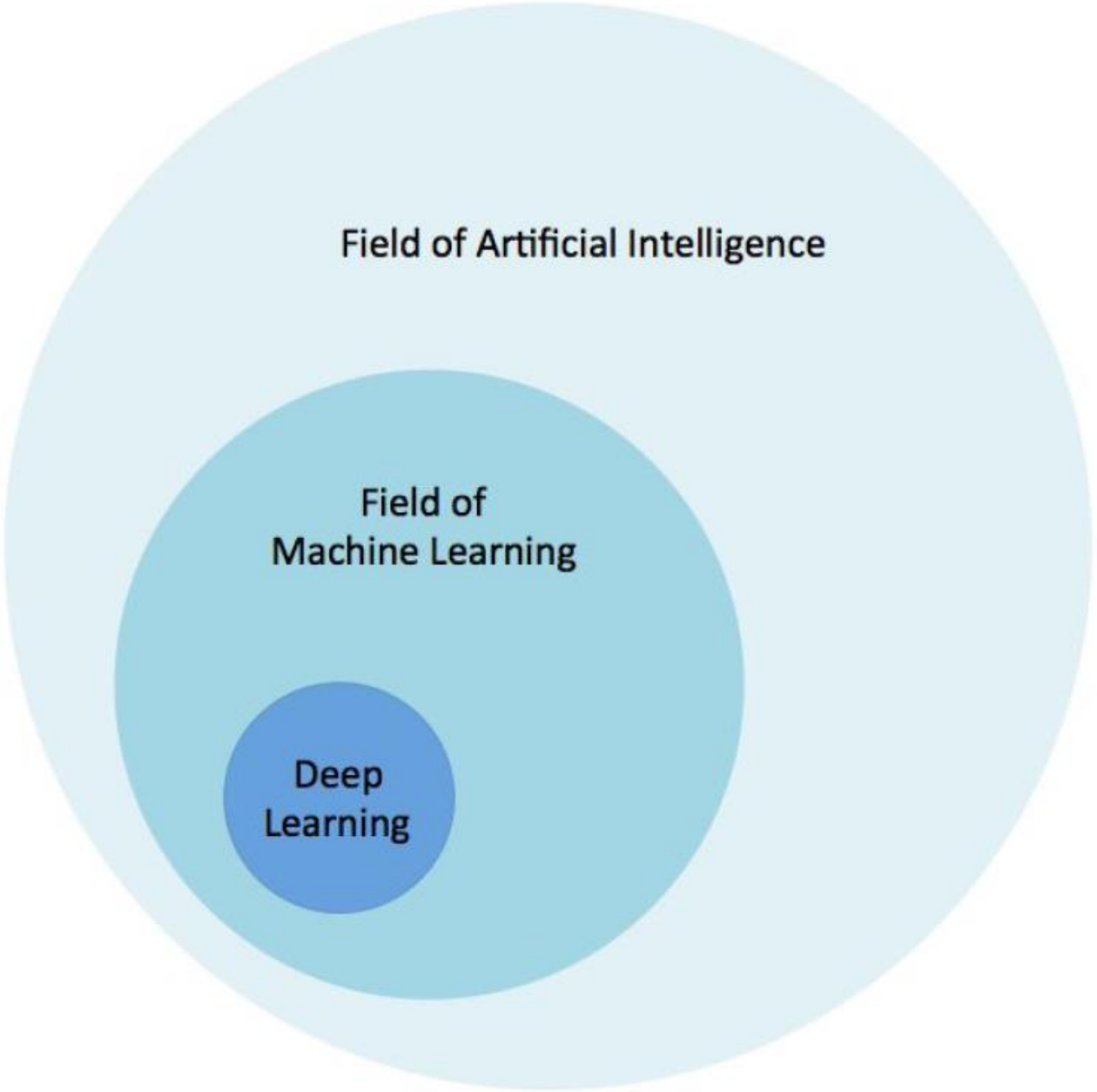
Background



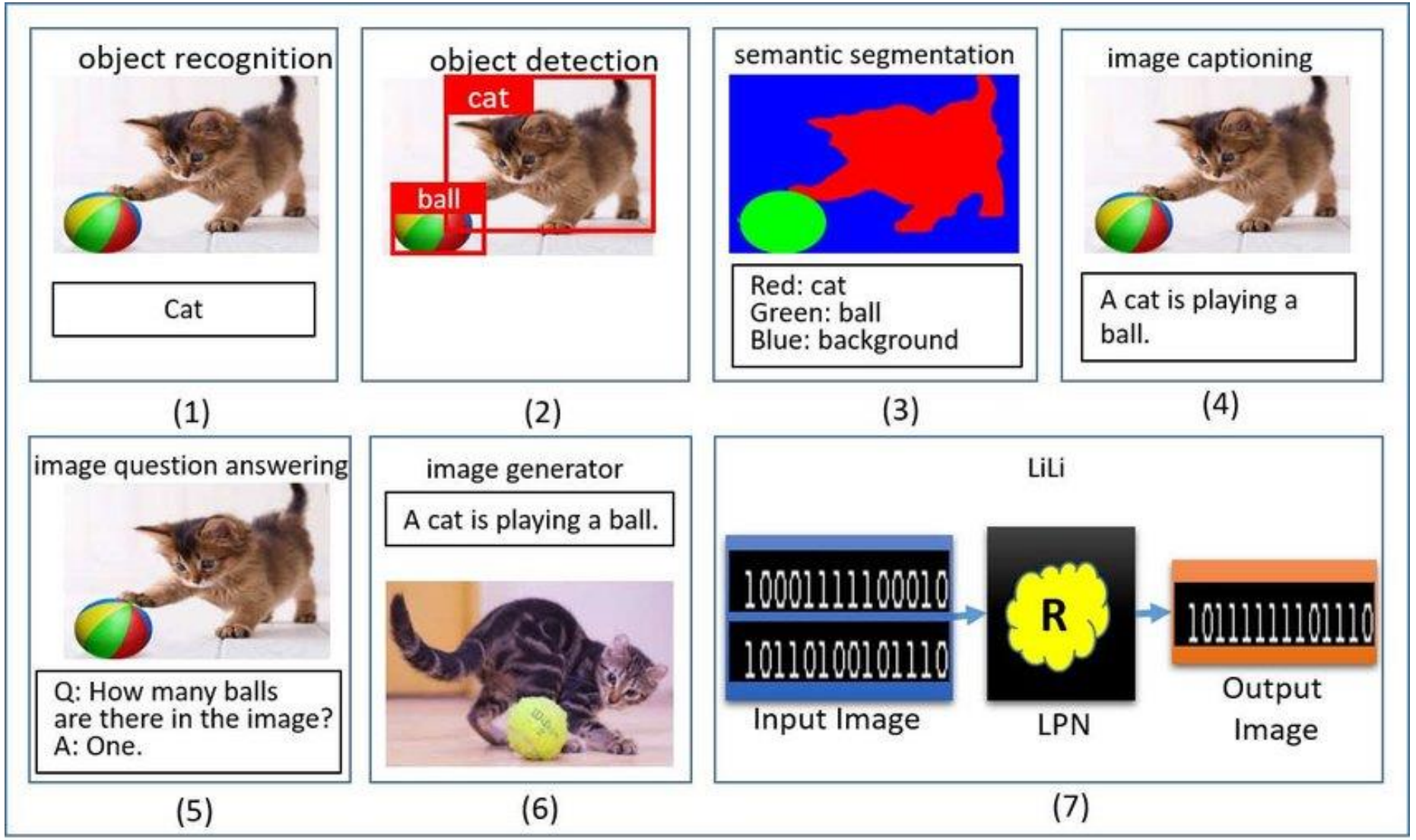
Группа компаний Б1 и ассоциация «ИнтерАгроТех» провели опрос представителей агробизнеса на тему кадрового голода (всего опросили 172 компании в сфере растениеводства).

- 88% опрошенных заявили, что ситуация с кадрами ухудшается
- 80% респондентов считают наиболее востребованными и дефицитными специальностями механизатора-тракториста и механизатора-комбайнера. Также заметно не хватает водителей грузовиков и спецтехники, слесарей-механиков и агрономов
- 68% респондентов заявили, что из-за нехватки рабочих рук их сотрудники работают с повышенной нагрузкой
- 30% респондентов внедряют средства автоматизации и цифровизации для решения проблемы
- 94% респондентов повышают зарплаты и премии для удержания сотрудников
- Недостаток кадров сильнее всего влияет на проведение уборочной кампании (по мнению 87% опрошенных), механическую обработку почвы (64%) и посевную кампанию (62%)

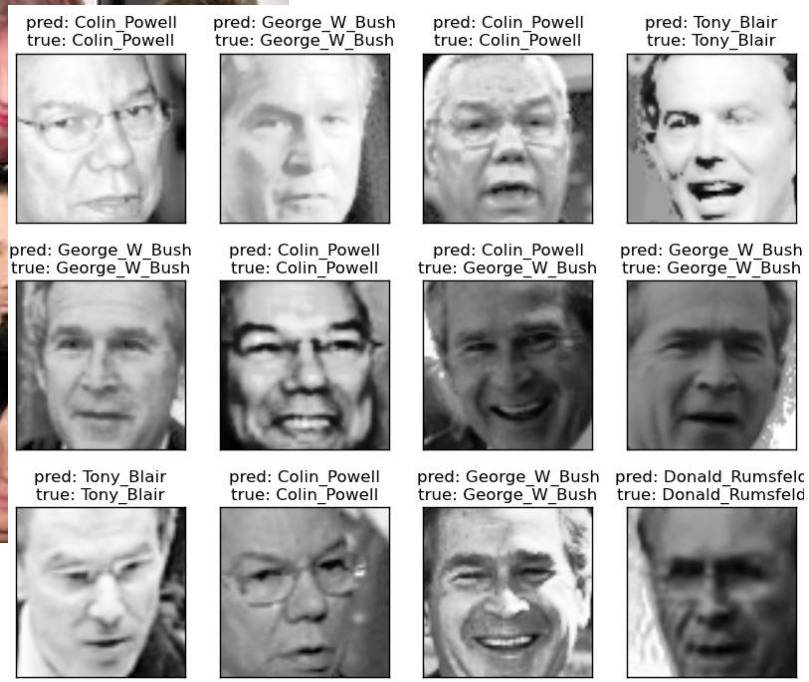
Basis - terms


Field of Artificial Intelligence
Field of
Machine Learning
Deep
Learning

Computer vision tasks

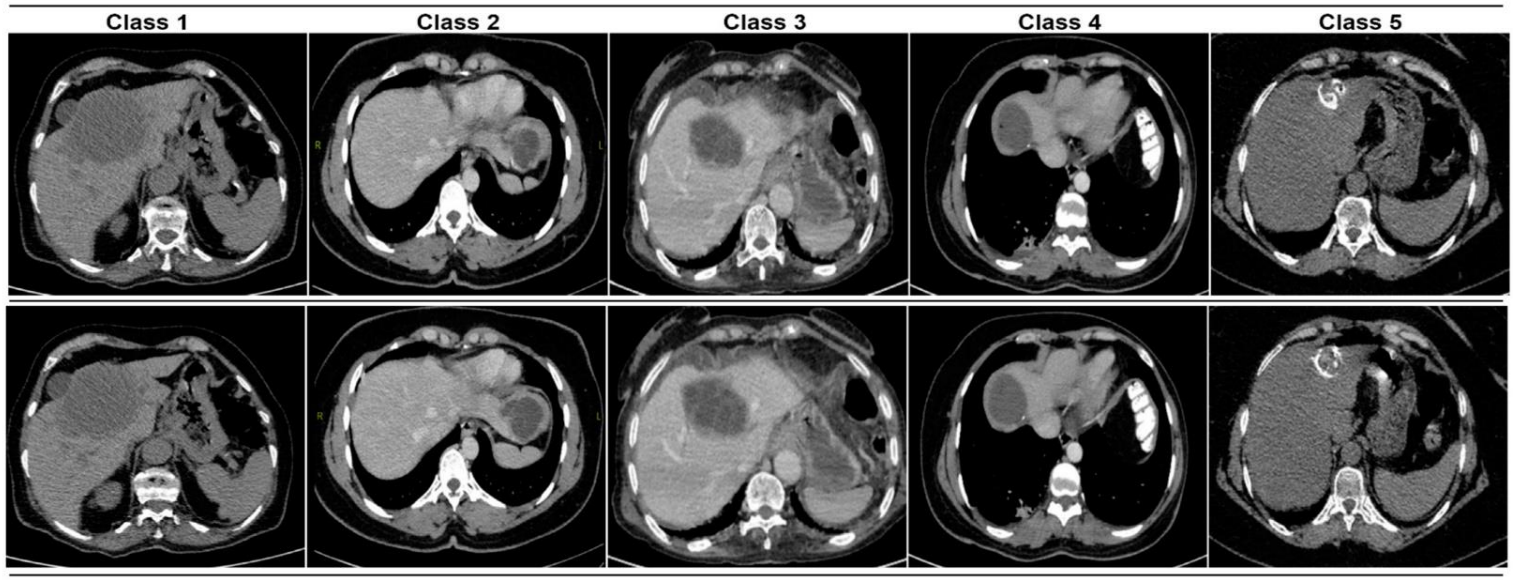


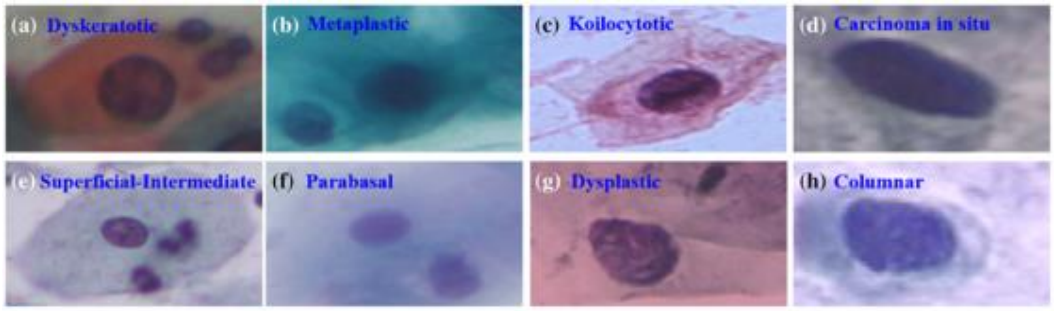
Point of image classification

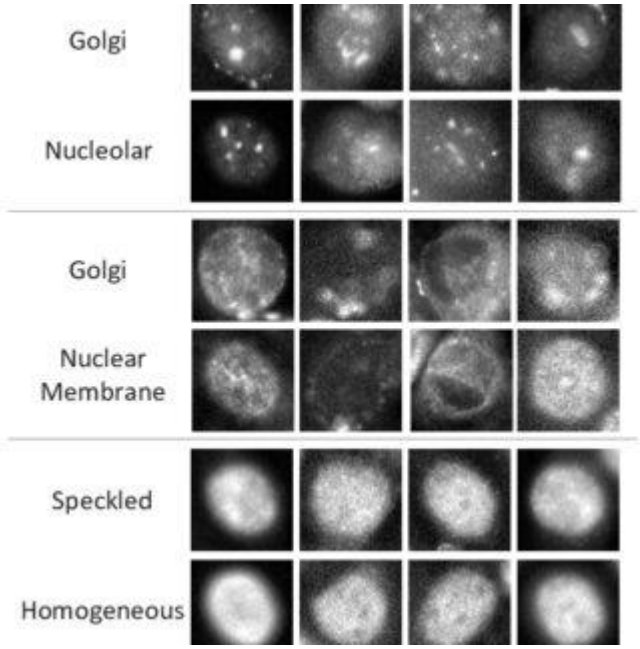


Point of classification

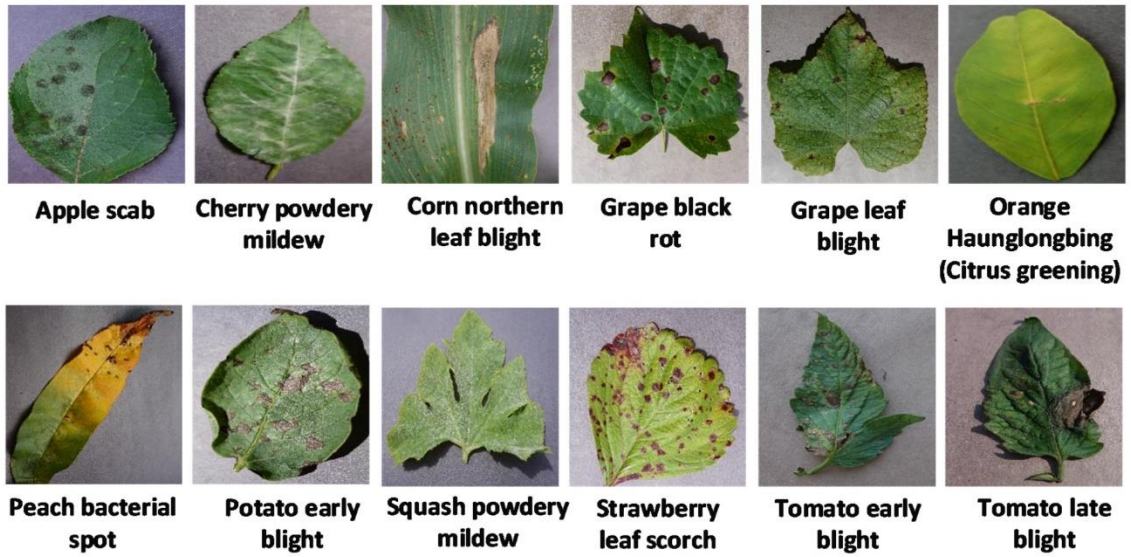
Point of classification

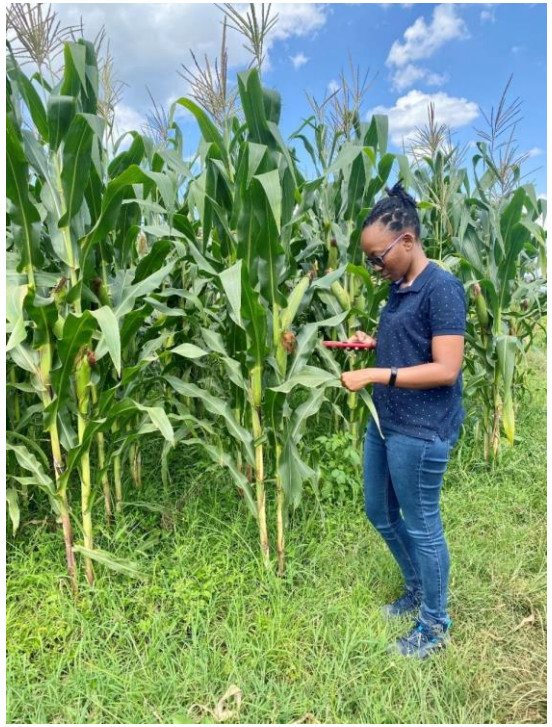


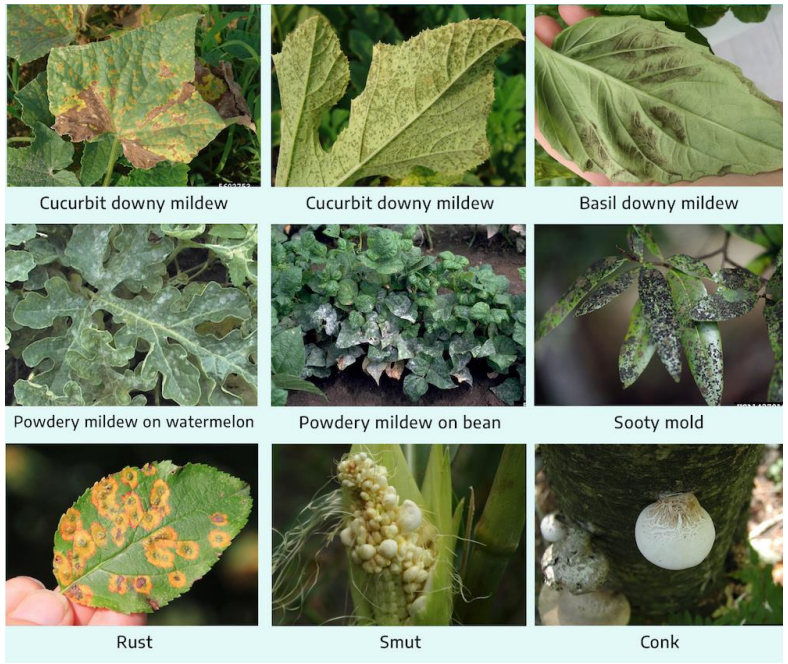




Point of classification



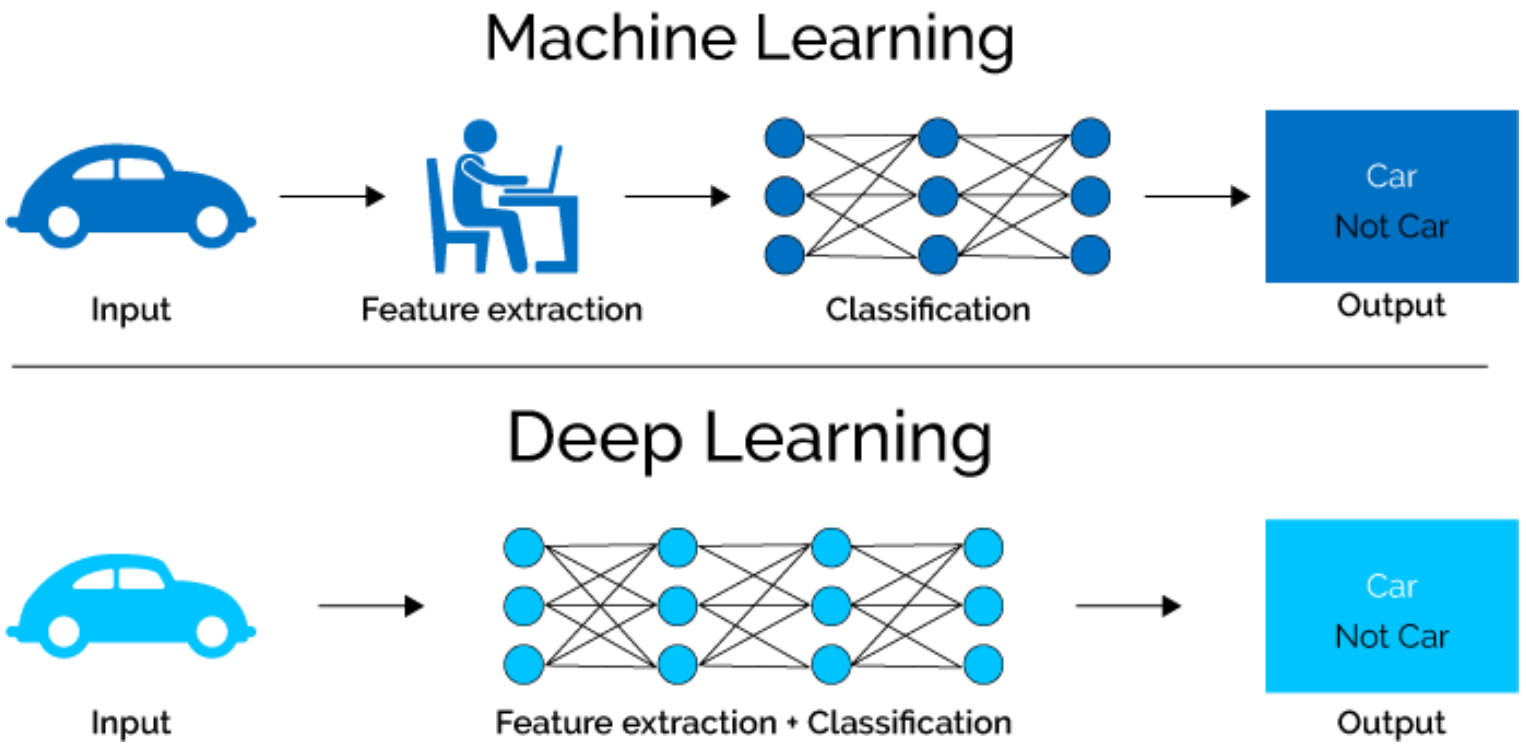




Basis - ML vs. Deep Learning

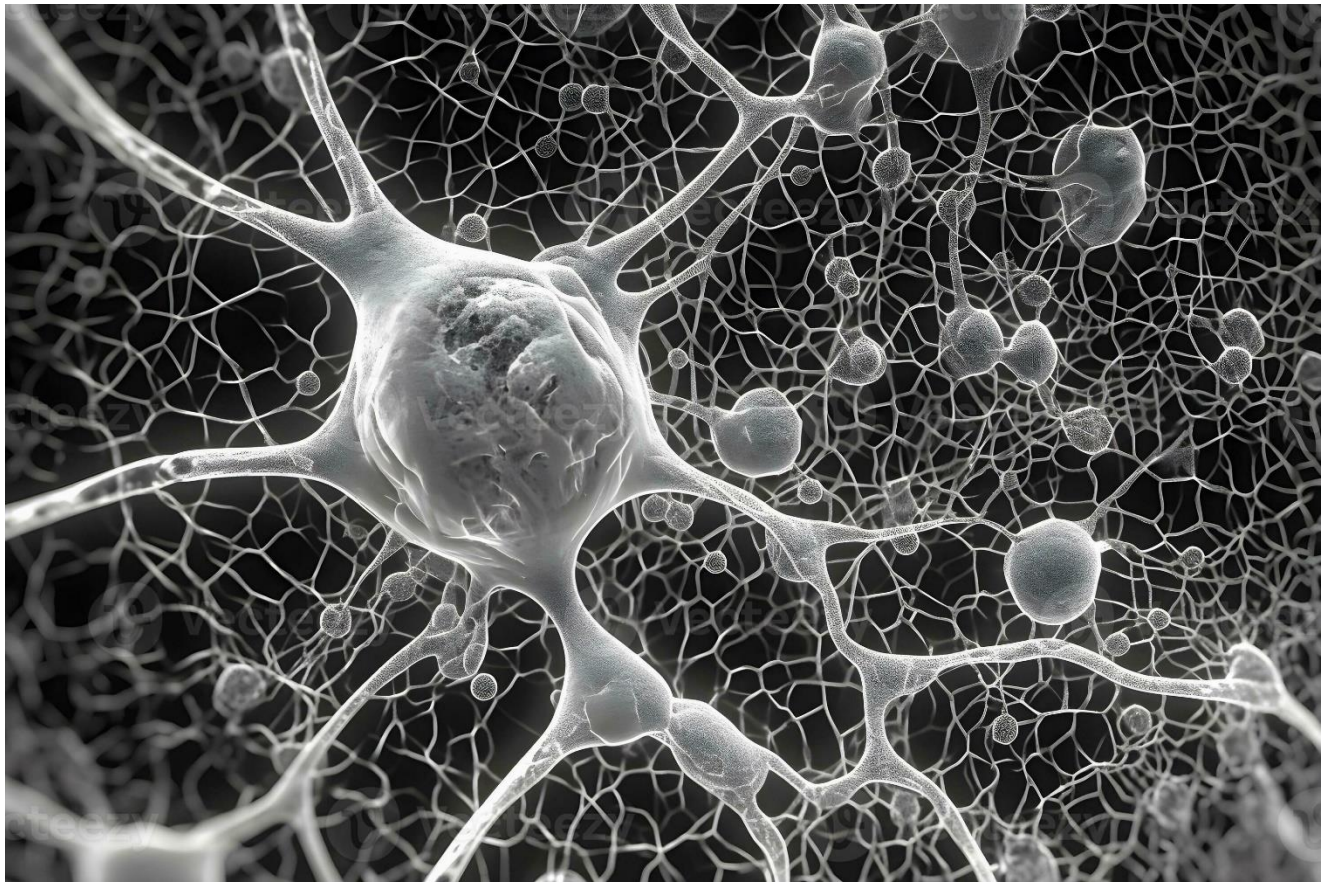
Deep learning (DL) is a machine learning subfield that uses multiple layers for learning data representations

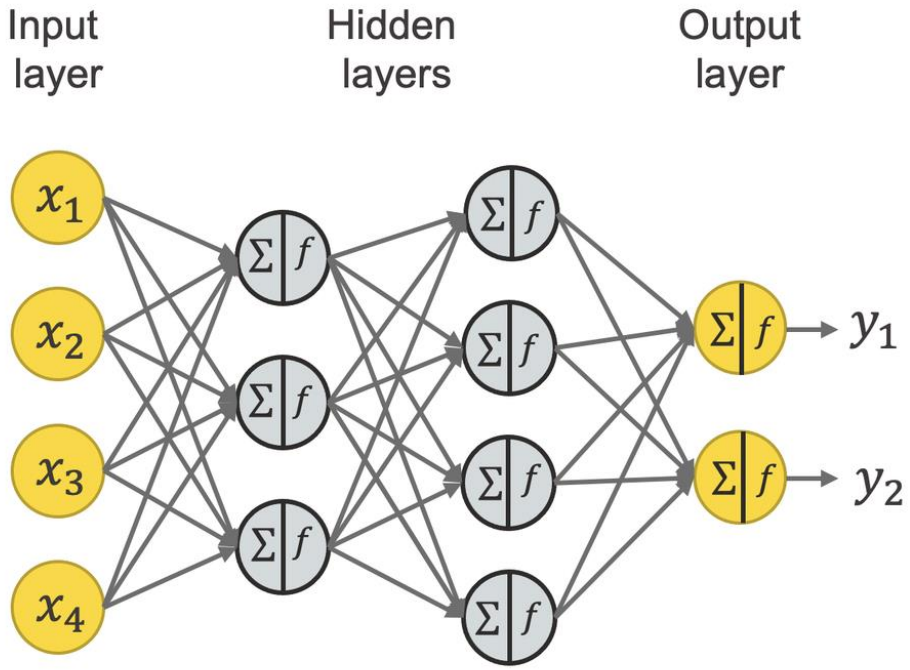
DL is exceptionally effective at learning patterns



Neural networks

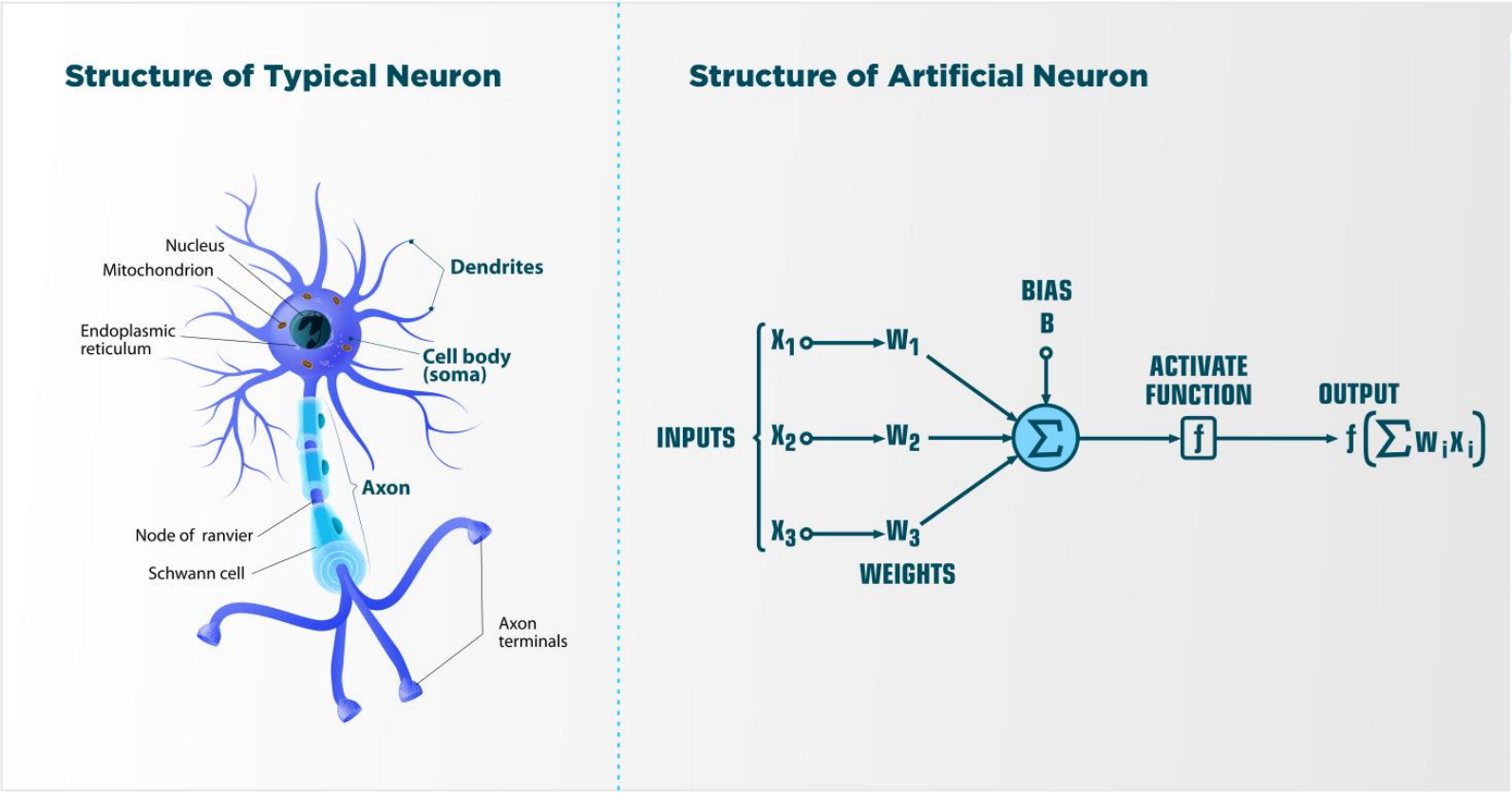
Neural networks are an example of a supervised learning algorithm and seek to approximate the function represented by your data

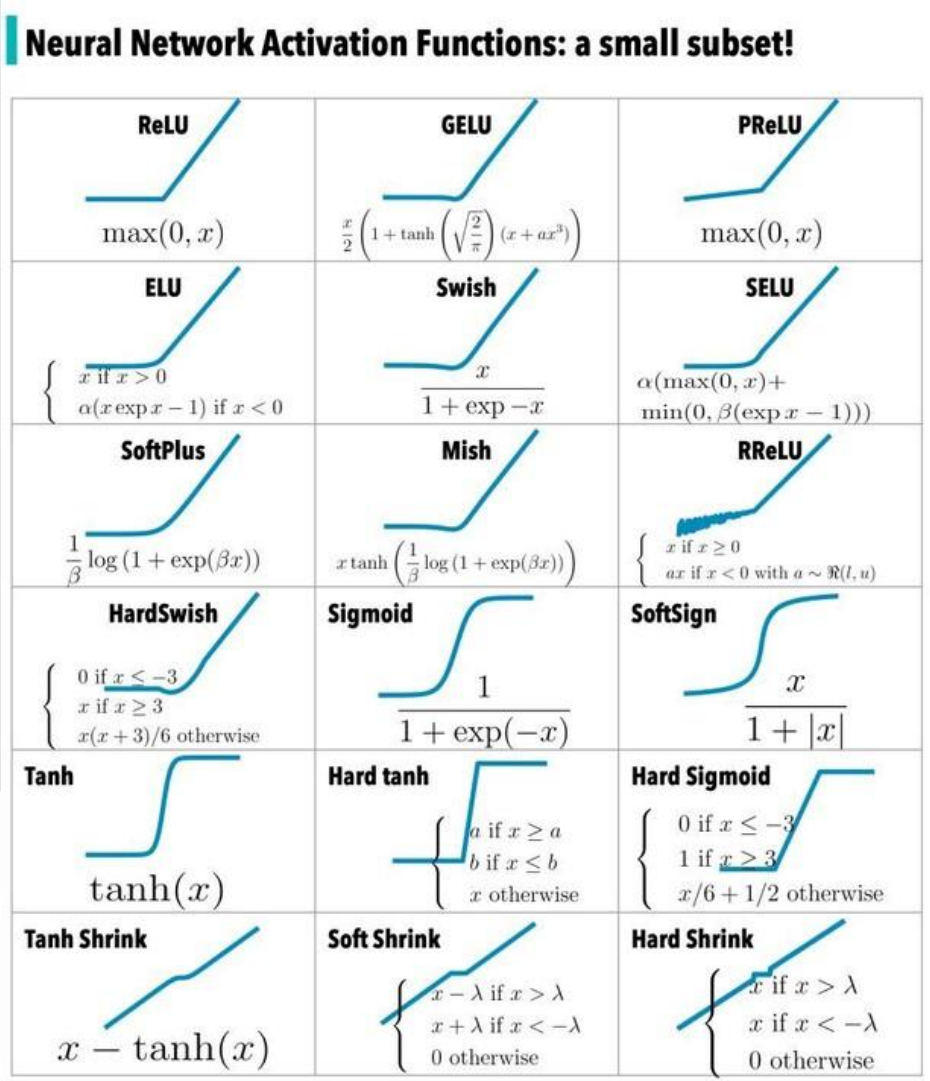




Farley and Wesley A. Clark were the first to simulate a Hebbian network in 1954 at MIT. They used computational machines, then called "calculators".

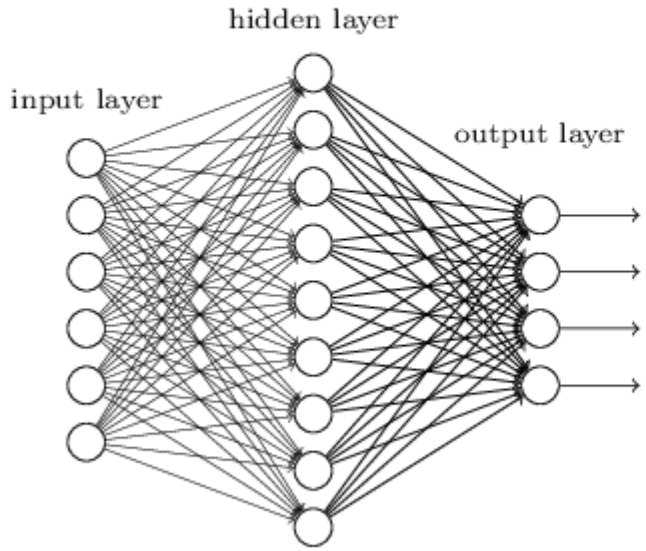
Neural networks



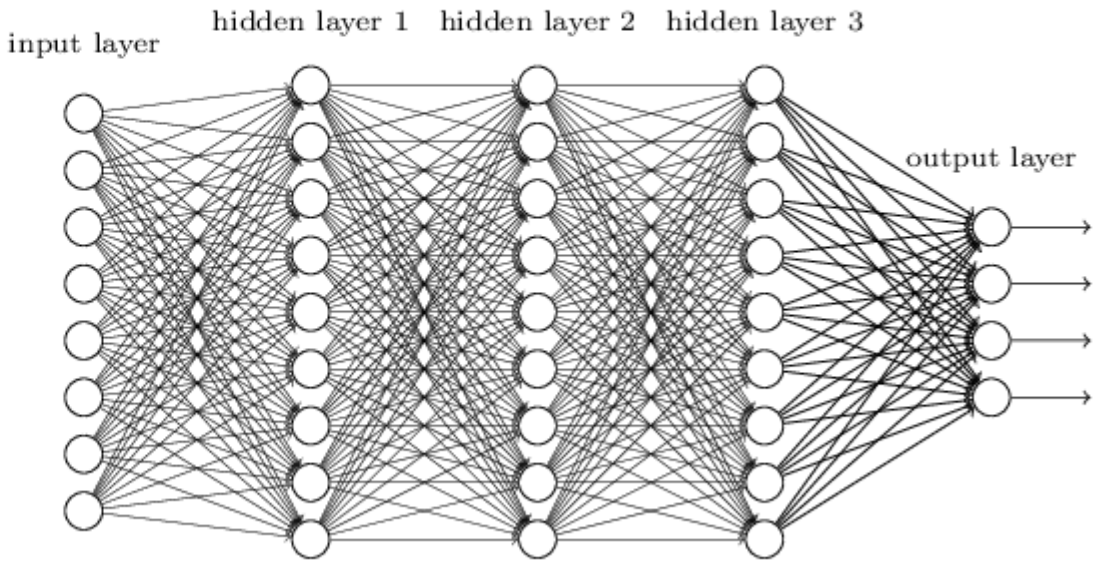


Deep neural networks

"Non-deep" feedforward neural network

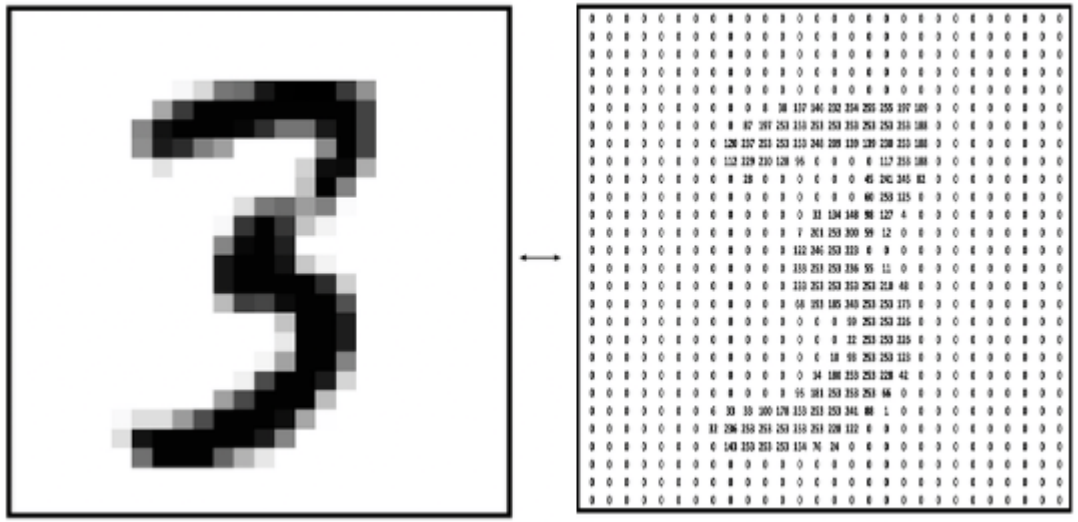


Deep neural network



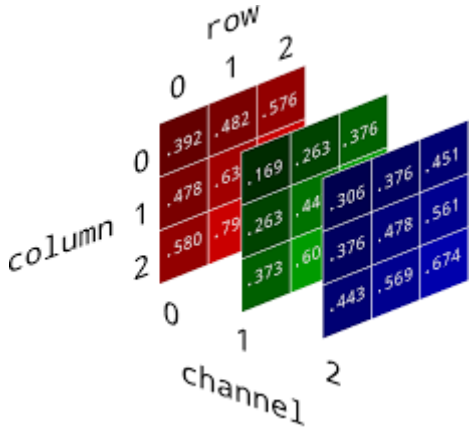
A neural network with multiple hidden layers and multiple nodes in each hidden layer is known as a **deep learning system** or a **deep neural network**. Deep learning is the development of deep learning algorithms that can be used to train and predict output from complex data.

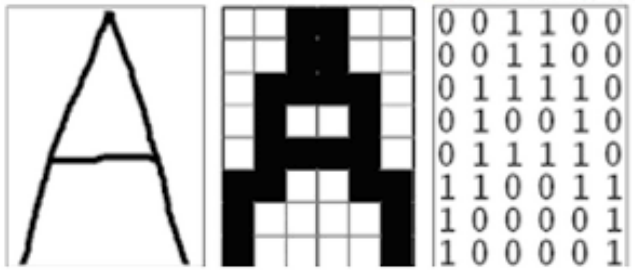
What computer sees?



What we see

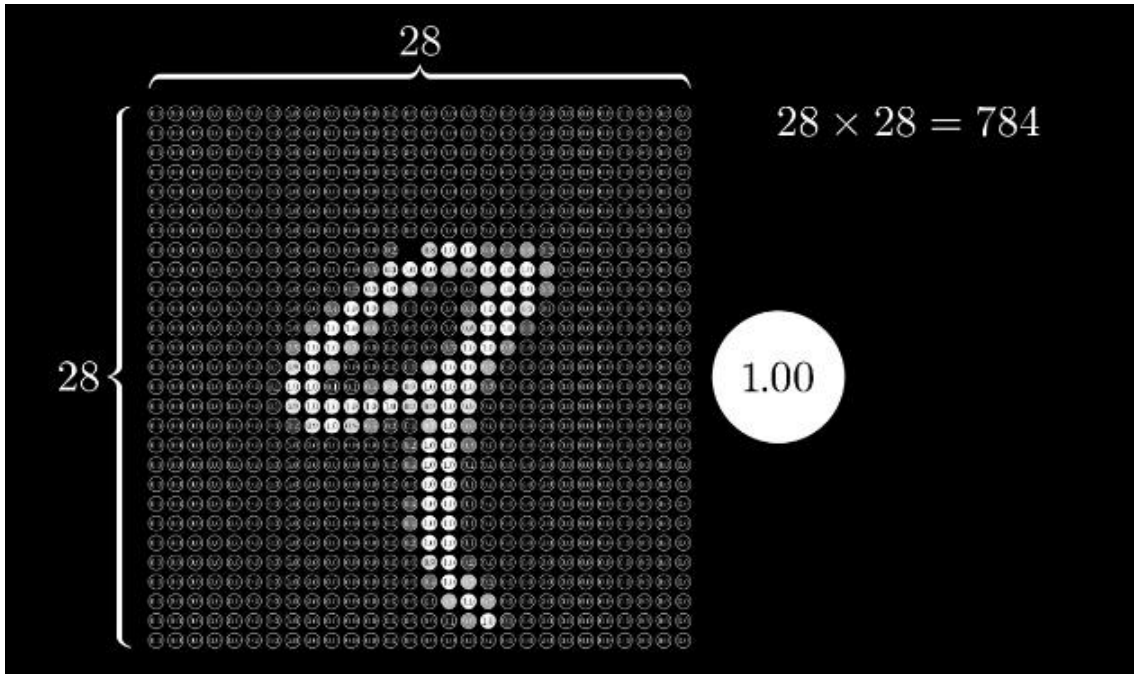
What a computer sees

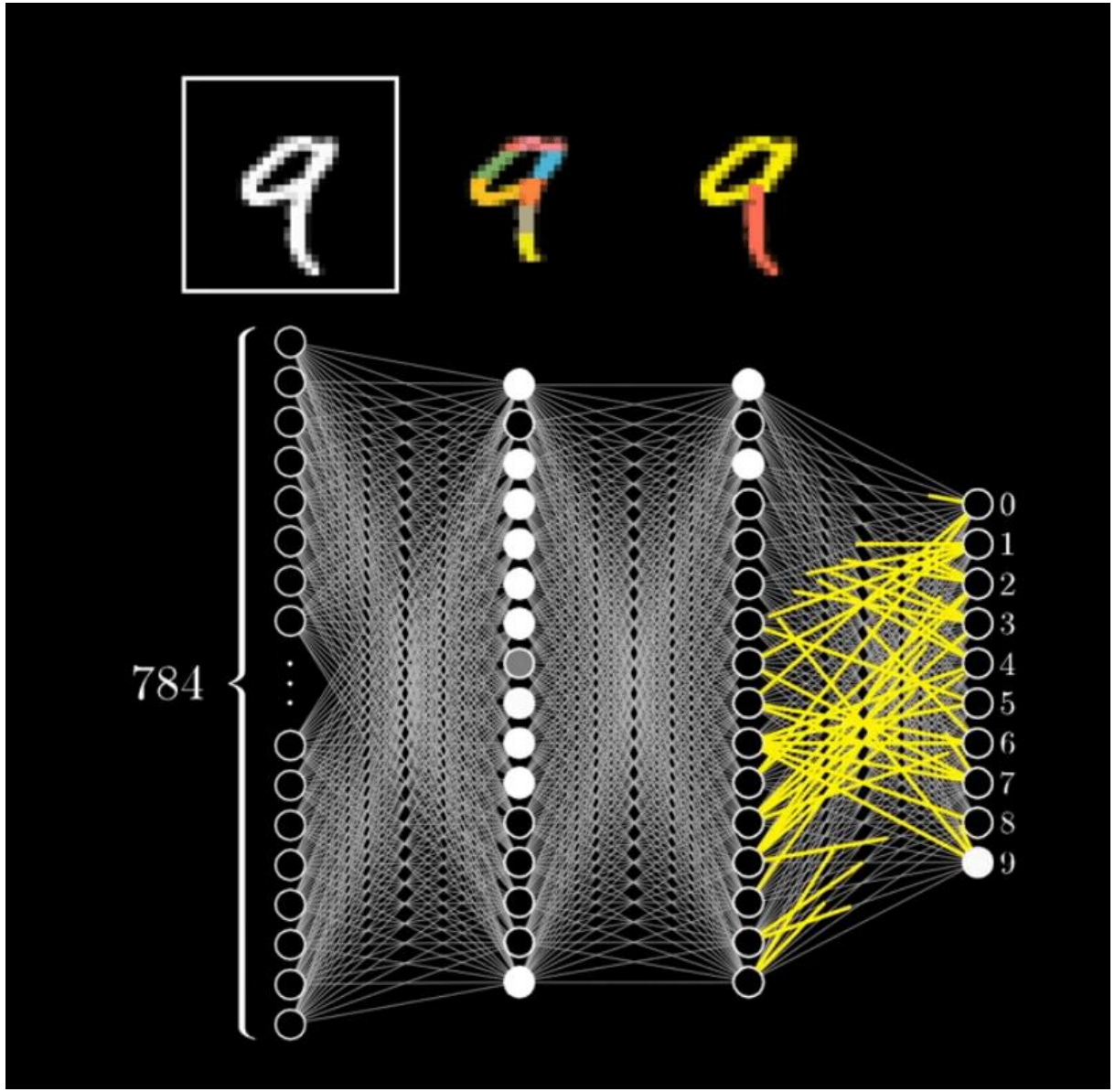




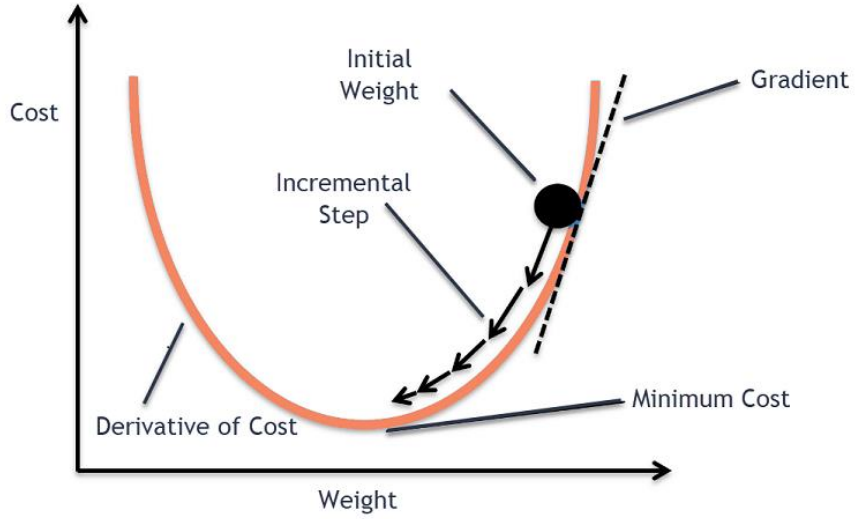
The image itself is a source of information.

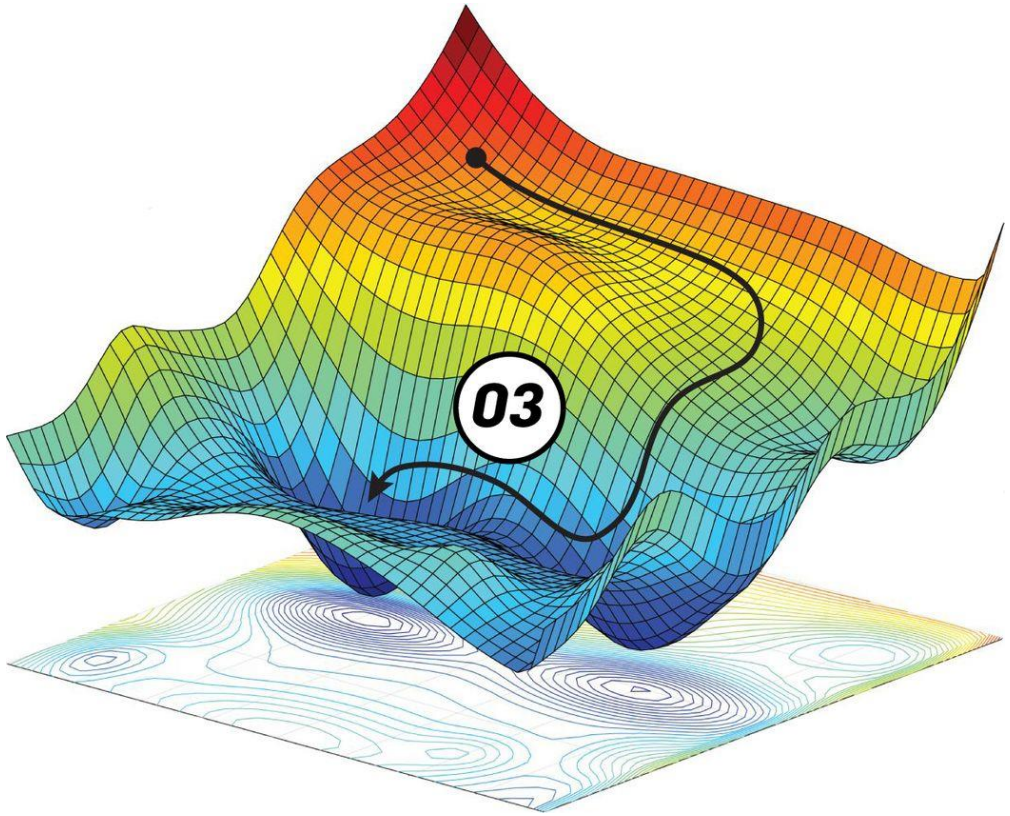
Classification of handwritten digits





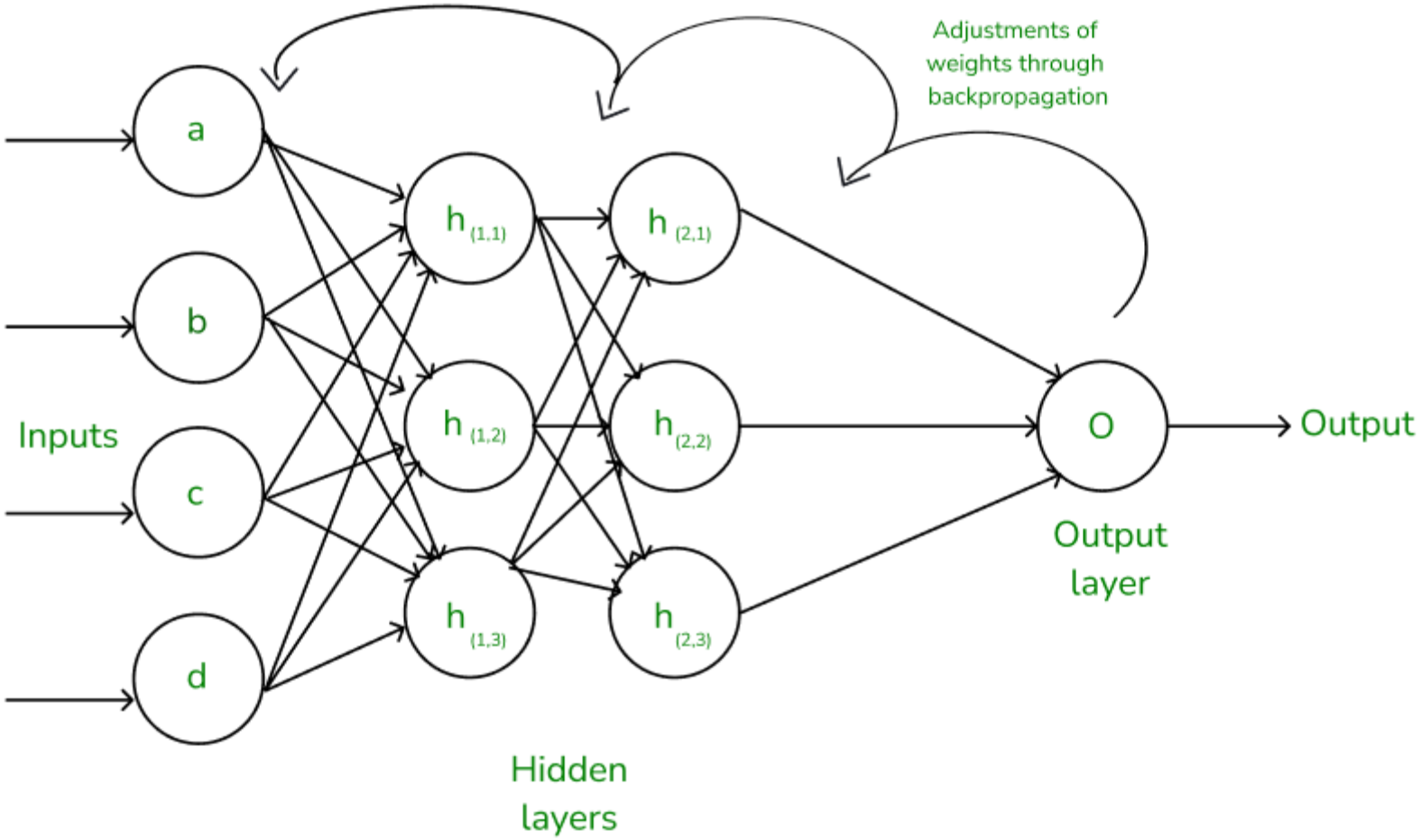
Basis - gradient descent





Gradient descent is an optimization algorithm which is commonly-used to train machine learning models and neural networks. It trains machine learning models by minimizing errors between predicted and actual results.

Basis - backpropaqation



In machine learning, backpropagation is a gradient estimation method used to train neural network models. The gradient estimate is used by the optimization algorithm to compute the network parameter updates.

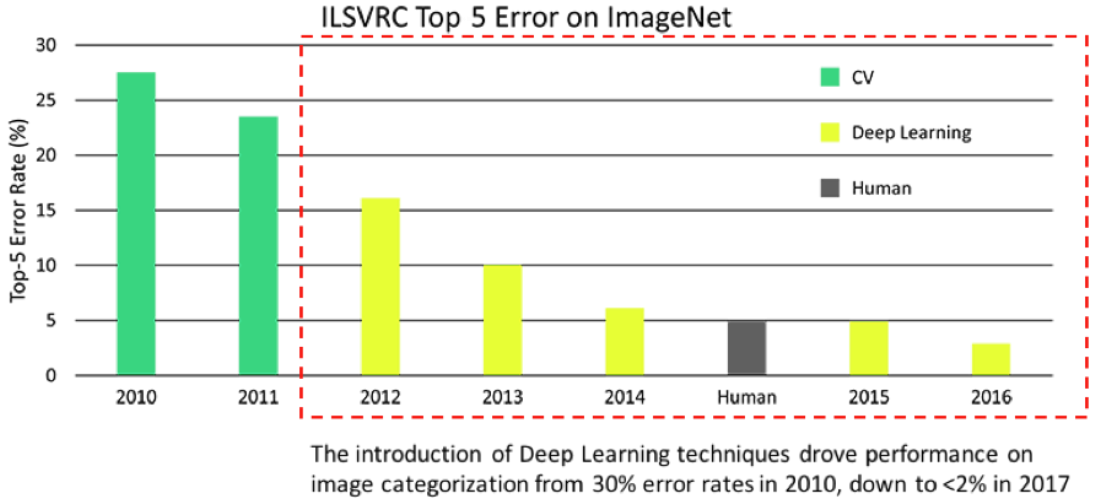
ImageNet

This dataset spans 1000 object classes and contains 1,281,167 training images, 50,000 validation images and 100,000 test images.

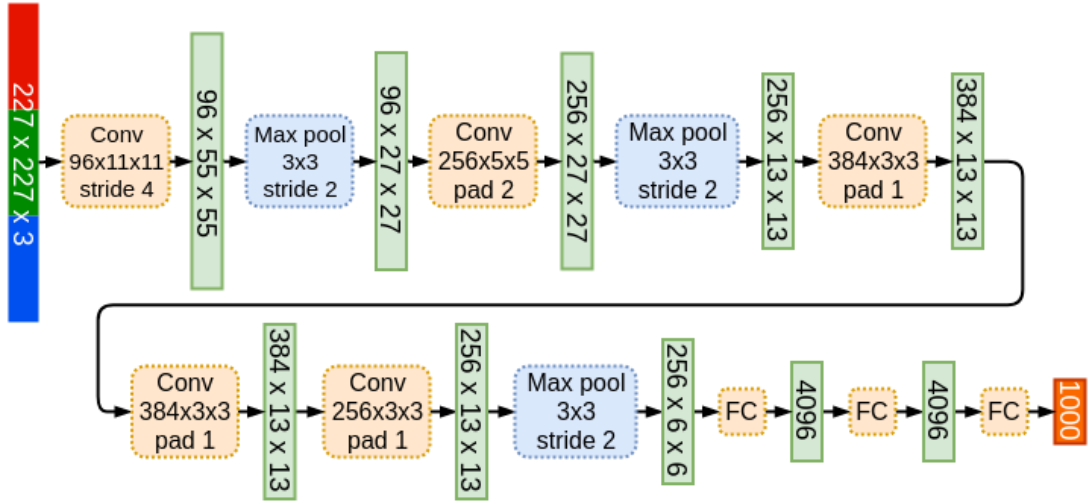
Since 2010 the dataset is used in the ImageNet Large Scale Visual Recognition Challenge (ILSVRC), a benchmark in image classification and object detection.

Deng J.; Dong W.; Socher R.; Li L.J., Li K.; Fei-Fei L. ImageNet: A large-scale hierarchical image database, 2009 IEEE Conference on Computer Vision and Pattern Recognition, Miami, FL, USA, 2009, pp. 248-255, doi: 10.1109/CVPR.2009.5206848.

AlexNet

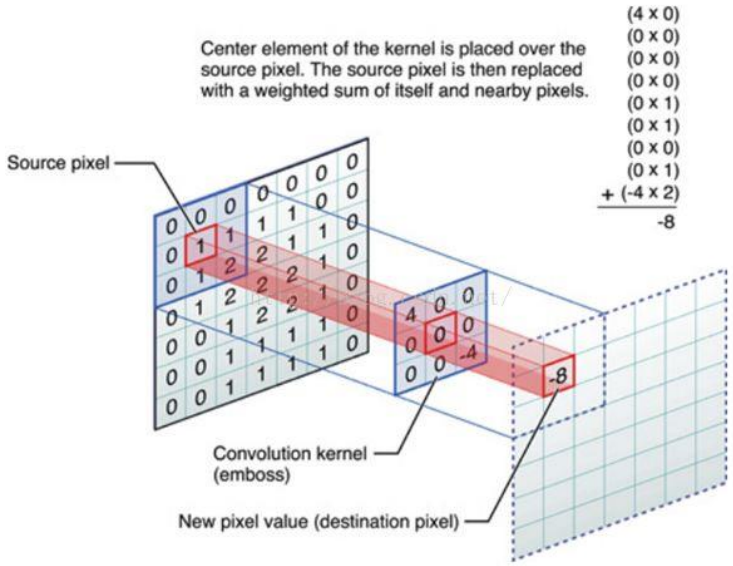


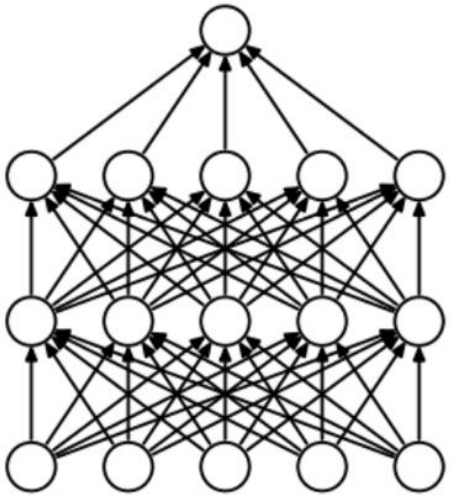
AlexNet is the winner of the 2012 **ImageNet Large Scale Visual Recognition Challenge** (**ILSVRC**, or simply ImageNet)



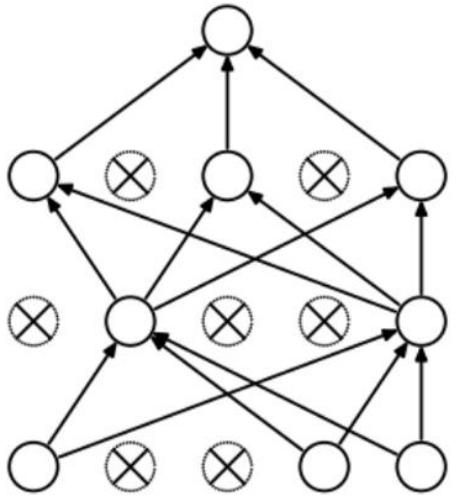
designed by Alex Krizhevsky in collaboration with Ilya Sutskever and Geoffrey Hinton, who was Krizhevsky's Ph.D. advisor at the University of Toronto

Basis - convolutional neural networks

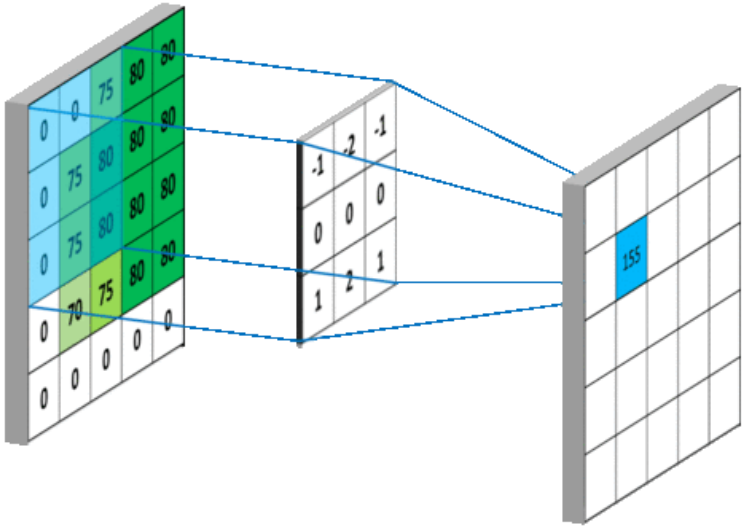


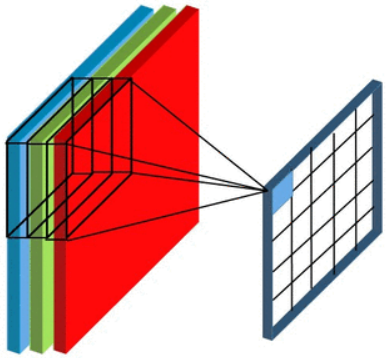


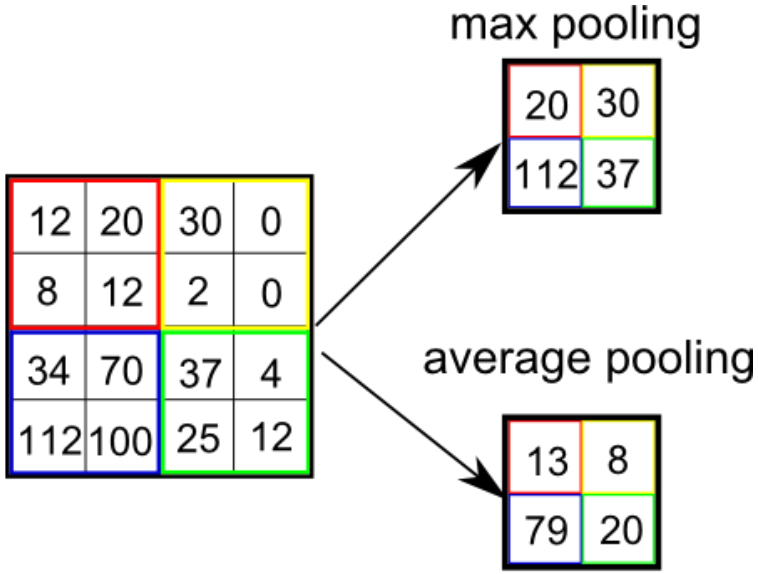
(a) Standard Neural Net



(b) After applying dropout.



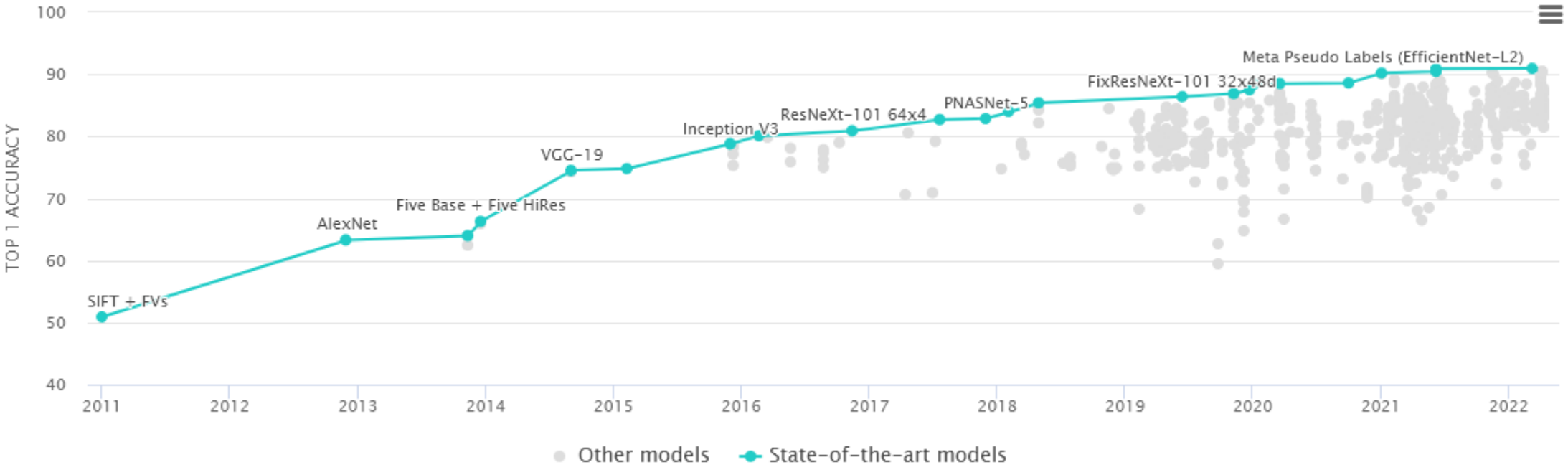




Basis – custom CNN architecture

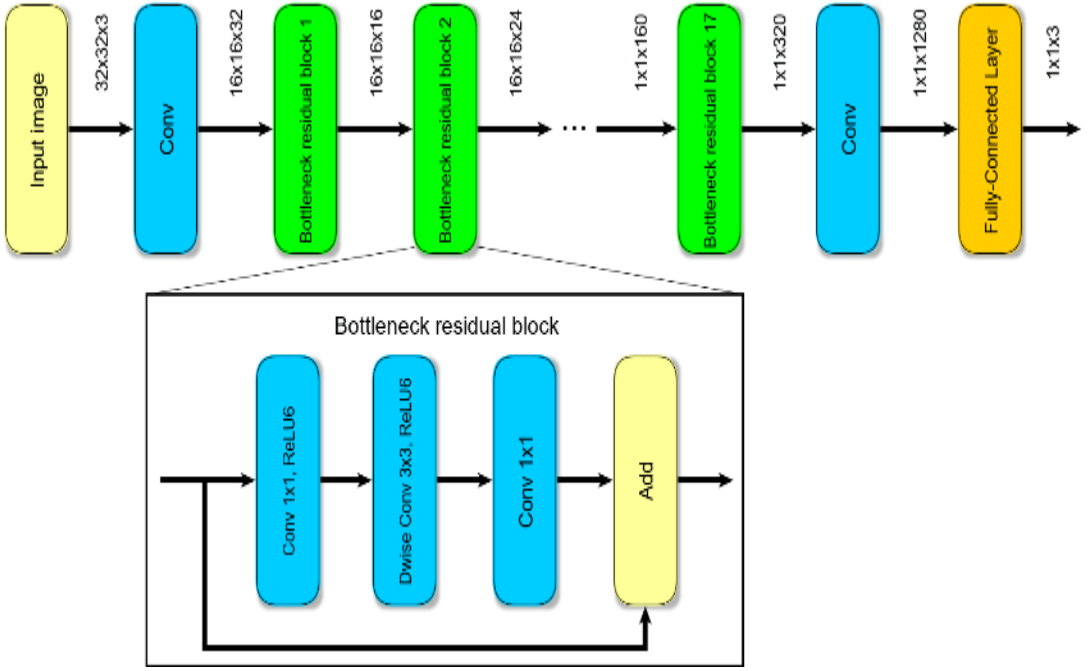

```python
def __init__(self, num_classes=10):
    super(CNNNet, self).__init__()
    self.features = nn.Sequential(
        nn.Conv2d(3, 64, kernel_size=11, stride=4, padding=2),
        nn.ReLU(),
        nn.MaxPool2d(kernel_size=3, stride=2),
        nn.Conv2d(64, 192, kernel_size=5, padding=2),
        nn.ReLU(),
        nn.MaxPool2d(kernel_size=3, stride=2),
        nn.Conv2d(192, 384, kernel_size=3, padding=1),
        nn.ReLU(),
        nn.Conv2d(384, 256, kernel_size=3, padding=1),
        nn.ReLU(),
        nn.Conv2d(256, 256, kernel_size=3, padding=1),
        nn.ReLU(),
        nn.MaxPool2d(kernel_size=3, stride=2),
    )
    self.avgpool = nn.AdaptiveAvgPool2d((6, 6))
    self.classifier = nn.Sequential(
        nn.Dropout(),
        nn.Linear(256 * 6 * 6, 4096),
        nn.ReLU(),
        nn.Dropout(),
        nn.Linear(4096, 4096),
        nn.ReLU(),
        nn.Linear(4096, num_classes)
    )

def forward(self, x):
    x = self.features(x)
    x = self.avgpool(x)
    x = torch.flatten(x, 1)
    x = self.classifier(x)
    return x
```

ImageNet- competitors



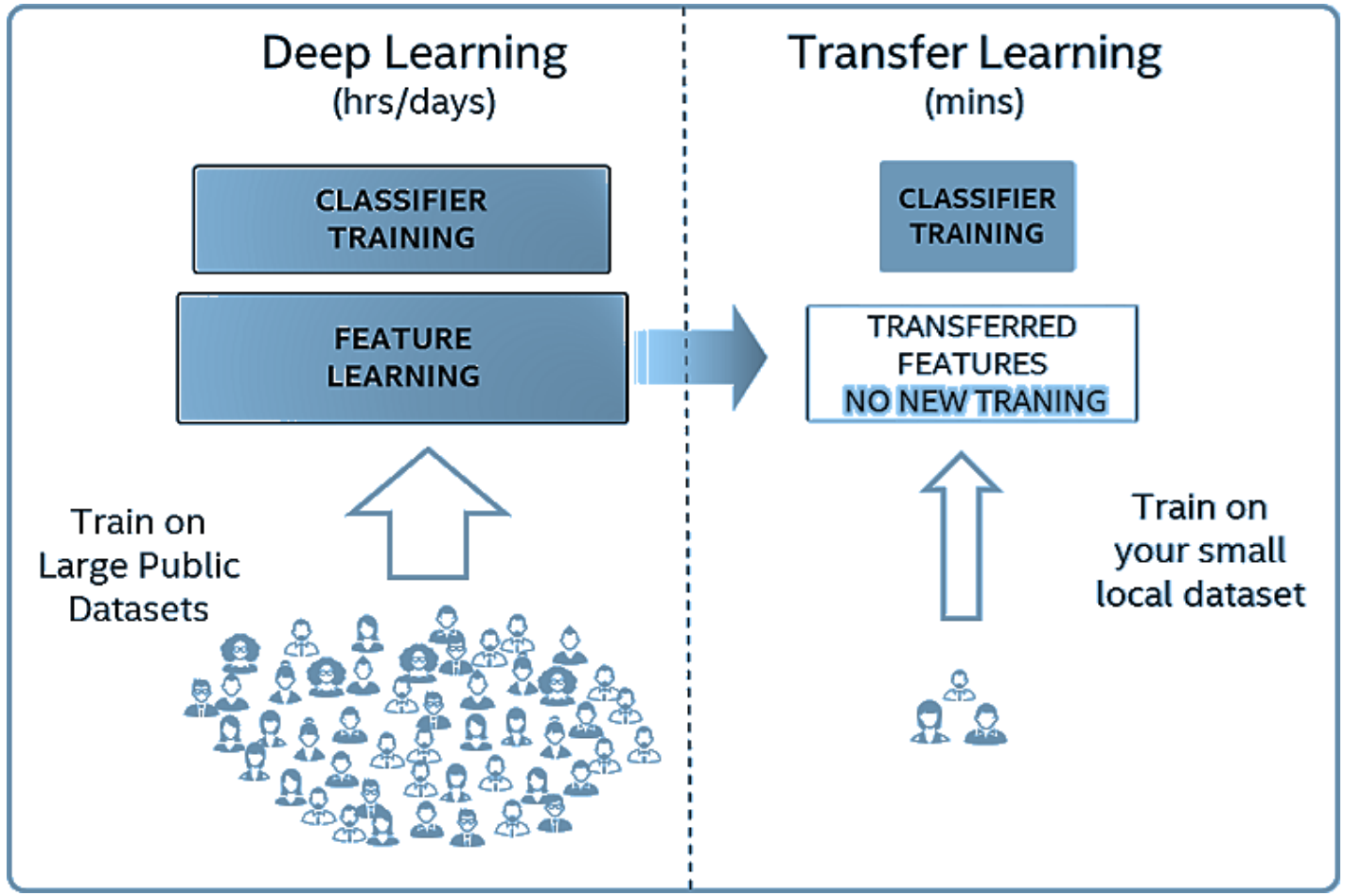
MobileNet_V2

```
MobileNetV2(
 (features): Sequential(
  (0): Conv2dNormActivation(
   (0): Conv2d(3, 32, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1), bias=False)
   (1): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
   (2): ReLU6(inplace=True)
  )
  (1): InvertedResidual(
   (conv): Sequential(
    (0): Conv2dNormActivation(
     (0): Conv2d(32, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), groups=32, bias=False)
     (1): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
     (2): ReLU6(inplace=True)
    )
    (1): Conv2d(32, 16, kernel_size=(1, 1), stride=(1, 1), bias=False)
    (2): BatchNorm2d(16, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
   )
  )
  (2): InvertedResidual(
   (conv): Sequential(
    (0): Conv2dNormActivation(
     (0): Conv2d(16, 96, kernel_size=(1, 1), stride=(1, 1), bias=False)
     (1): BatchNorm2d(96, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
     (2): ReLU6(inplace=True)
    )
    (1): Conv2dNormActivation(
     (0): Conv2d(96, 96, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1), groups=96, bias=False)
     (1): BatchNorm2d(96, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
     (2): ReLU6(inplace=True)
    )
    (2): Conv2d(96, 24, kernel_size=(1, 1), stride=(1, 1), bias=False)
    (3): BatchNorm2d(24, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
   )
  )

...

  (18): Conv2dNormActivation(
   (0): Conv2d(320, 1280, kernel_size=(1, 1), stride=(1, 1), bias=False)
   (1): BatchNorm2d(1280, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
   (2): ReLU6(inplace=True)
  )
 )
 (classifier): Sequential(
  (0): Dropout(p=0.2, inplace=False)
  (1): Linear(in_features=1280, out_features=1000, bias=True)
 )
)
```



Basis – Transfer learning

How to learn if lack of data
Transfer Learning



- Find a deep neural network pretrained on a big dataset
- Replace the classification layer with a layer appropriate for your task
- Finetune the new classifier on specific data
- Voila! Use the new model for inference

Что если данных для обучения мало?

Few-shot Learning:

Few-shot learning (FSL) is a branch of supervised machine learning, focused on learning from limited number of examples

Inspired by human/animal ability to rapidly generalize from few examples.

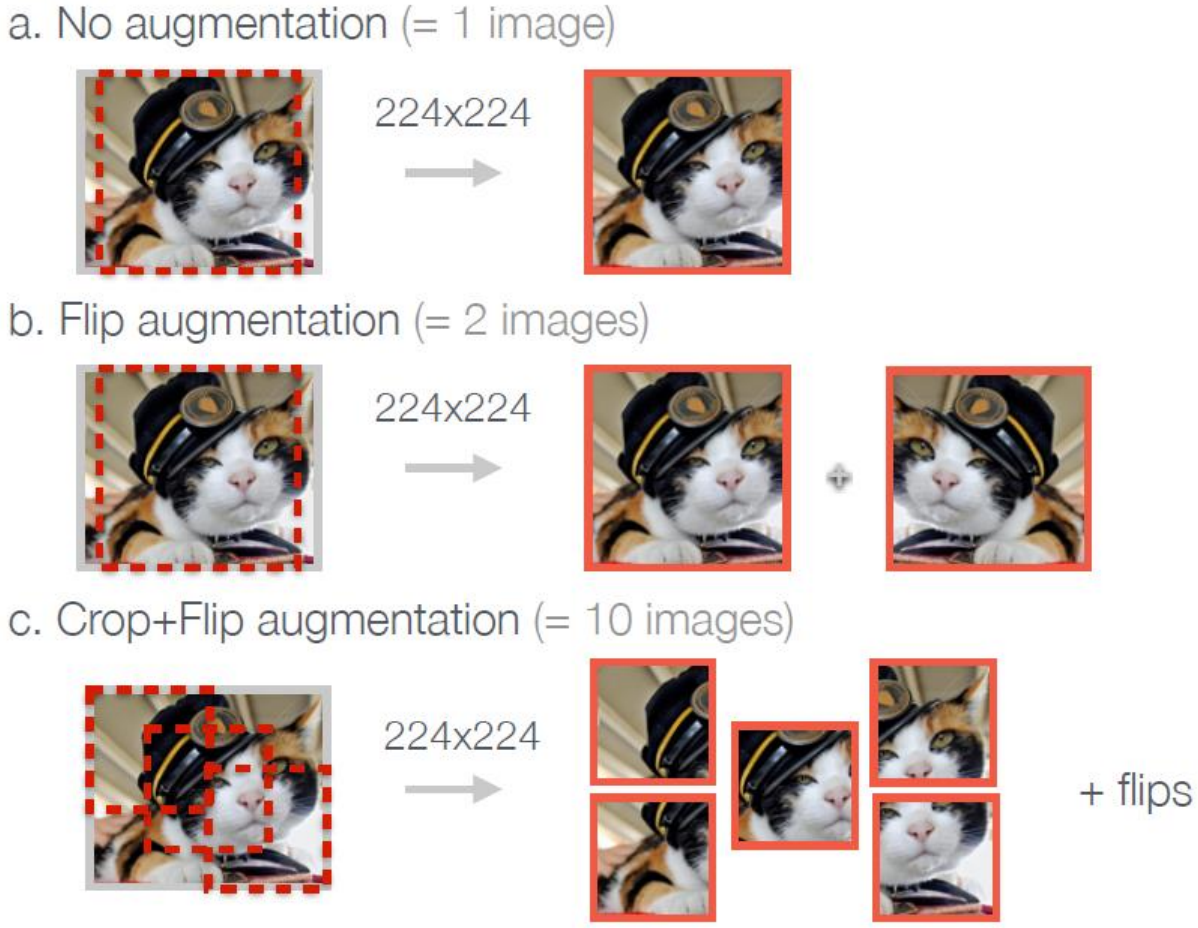
Typical scenarios[1]:
- Learning for rare cases – when obtaining labelled data is hard or impossible
- Reducing data gathering effort and computational cost

Variations:
- (<50, e.g. 5,10)-shot learning: General case, where only few examples are given
- 1-shot learning: Extreme case, where only one data example is given (e.g. 1 image per class)
- 0-shot learning: Instead of image, we have description of new class
- 'Less than 1'-shot learning[2]: N classes, M examples, M<N, use soft-labels

Why does it work?
In ML, experience is gained by fitting data.
In FSL we also have prior knowledge[1]. Two training phases:
1. Gain prior knowledge by fitting large amount of examples that are similar to goal task
2. Gain experience on small dataset for goal task.

Meta-Learning

Transformers, synthetic images

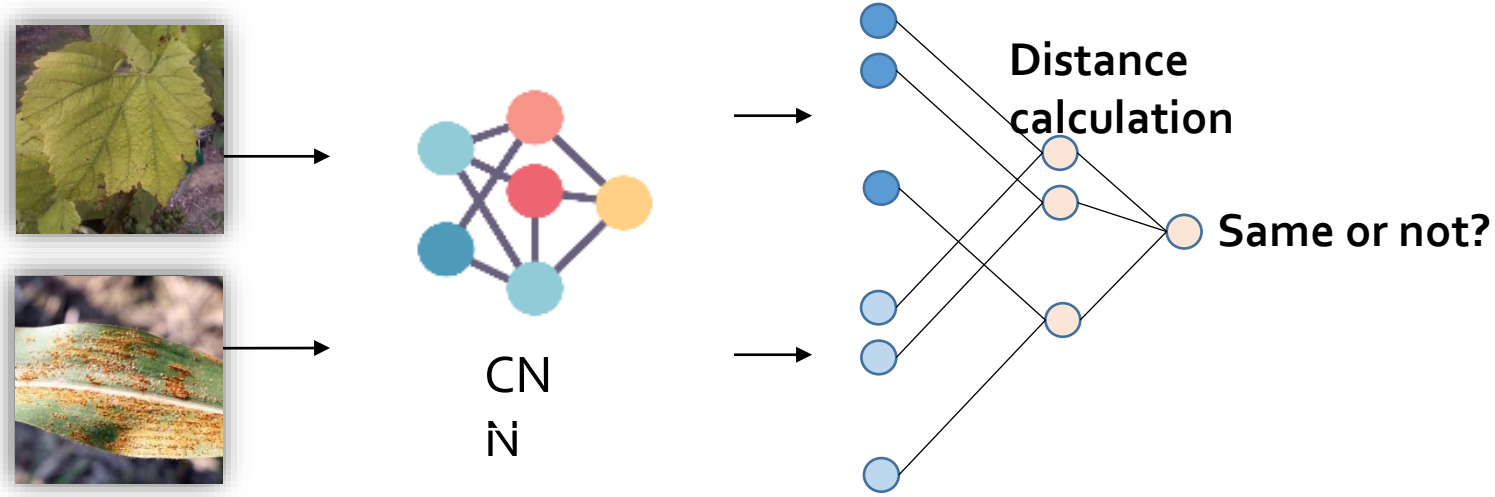


Augmentation

Multi-scale feature fusion, attention mechanism, self supervised, masked autoencoder and many other methods have been applied to the task self-supervised

Similarity learning – Siamese networks

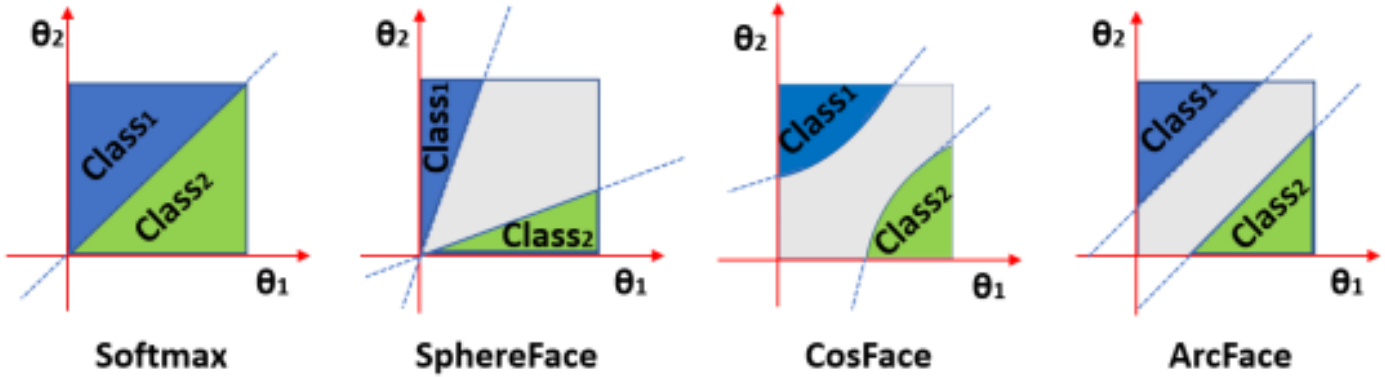
Siamese networks is a part of **one-shot learning** approach. One shot-learning aims to learn information about object categories from one, or only a few, training samples/images

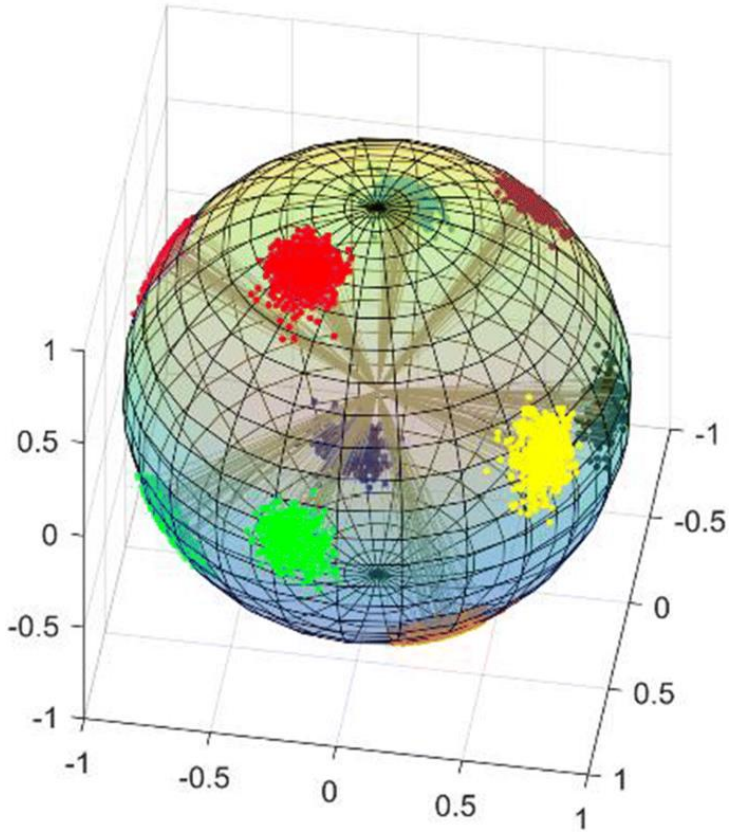


- Train siamese network on your own paired data

- Take the trained twin and append a classifier on top of it
- Finetune the new classifier on specific data appropriate for your task

Similarity learning – angular loss functions





ArcFace, or **Additive Angular Margin Loss**, is a loss function used in face recognition tasks. The softmax is traditionally used in these tasks. However, the softmax loss function does not explicitly optimise the feature embedding to enforce higher similarity for intraclass samples and diversity for inter-class samples, which results in a performance gap for deep face recognition under large intra-class appearance variations.

Эксперимент

Dataset

There are 68 classes without connection to the crop in the dataset, namely: Anthocyanosis, Anthracnose, Ants, Aphid, Aphid effects, Ascochyta blight, Bacterial spot, Black chaff, Black rot, Black spots, Blossom end rot, Botrytis cinerea, Burn, Canker, Caterpillars, Cherry leaf spot, Coccomyces of pome fruits, Colorado beetle, Colorado beetle effects, Corn downy mildew, Cyclamen mite, Downy mildew, Dry rot, Edema, Esca, Eyespot, Frost cracks, Galls, Grey mold, Gryllotalpa, Gryllotalpa effects, Healthy, Late blight, Leaf curl, Leaf deformation, Leaf miners, Leaf spot, Leaves scorch, Lichen, Loss of foliage turgor, Marginal leaf necrosis, Mealybug, Mechanical damage, Monilia, Mosaic virus, Northern leaf blight, Nutrient deficiency, Pear blister mite, Pest damage, Polypore, Powdery mildew, Rust, Scab, Scale, Shot hole, Shute, Slugs, Slugs caterpillars effects, Sooty mold, Spider mite, Thrips, Tubercular necrosis, Verticillium wilt, Whitefly, Wilting, Wireworm, Wireworm effects, and Yellow leaves.

The minimum and maximum number of images per class are 20 and 202, respectively. The images are RGB and have a resolution of 256x256 pixels. Examples of these images are presented in the figure.

Normalization do we need it?

ImageNet:
mean = [0.485, 0.456, 0.406],
std = [0.229, 0.224, 0.225]

DoctorP:
mean=[0.4467, 0.4889, 0.3267],
std=[0.2299, 0.2224, 0.2289]

The ability of the models to adapt and generalize to new data is very important. Along with different training approaches, the generalization of models and the effect of image normalization will be investigated. An additional dataset of 150 images, gathered from hard-case user requests, will be used for this purpose.

Models zoo

PyTorch is an open source machine learning (ML) framework based on the Python programming language and the Torch library. Torch is an open source ML library used for creating deep neural networks and is written in the Lua scripting language. It's one of the preferred platforms for deep learning research. The framework is built to speed up the process between research prototyping and deployment.

	Acc@1 (on ImageNet-1K)	Model file size (Mb)	Num. of parameters
ConvNeXt_Small	82.52	191.7	50223688
EfficientNet_B3	82.008	47.2	12233232
Inception_V3	77.294	103.9	27161264
MNASNet1_3	76.506	24.2	6282256
MobileNetV2	71.878	13.6	3504872
MobileNet_V3_Large	74.042	21.1	5483032
RegNet_X_1_6GF	77.04	35.3	9190136
ResNet101	77.374	170.5	44549160
ResNeXt50_32X4D	77.618	95.8	25028904

The torchvision.models subpackage contains definitions of models for addressing different tasks, including: image classification, pixelwise semantic segmentation, object detection, instance segmentation, person keypoint detection, video classification, and optical flow.

Metrics

For model evaluation metrics, mean epoch processing time (MEPT), accuracy, and weighted average precision, recall, and F1-score were used. MEPT indicates the time needed to train a model for one epoch. Accuracy is the percentage of predictions that were made correctly. It can be formulated as follows:

$$\text{Accuracy} = \frac{TP+TN}{TP+FP+TN+FN}$$

where:

- TP is true positive,
- FP is false positive,
- TN is true negative,
- FN is false negative.

Weighted-average metrics take into account the number of samples of each class in the data. Formally, it is the sum of metrics multiplied by the weights of each class.

Basic recall, or true positive rate (TPR), is calculated as follows:

$$\text{Recall} = \frac{TP}{TP+FN}$$

Precision is calculated as follows:

$$\text{Precision} = \frac{TP}{TP+FP}$$

F1-score is the harmonic average of recall and precision and can be formulated as follows:

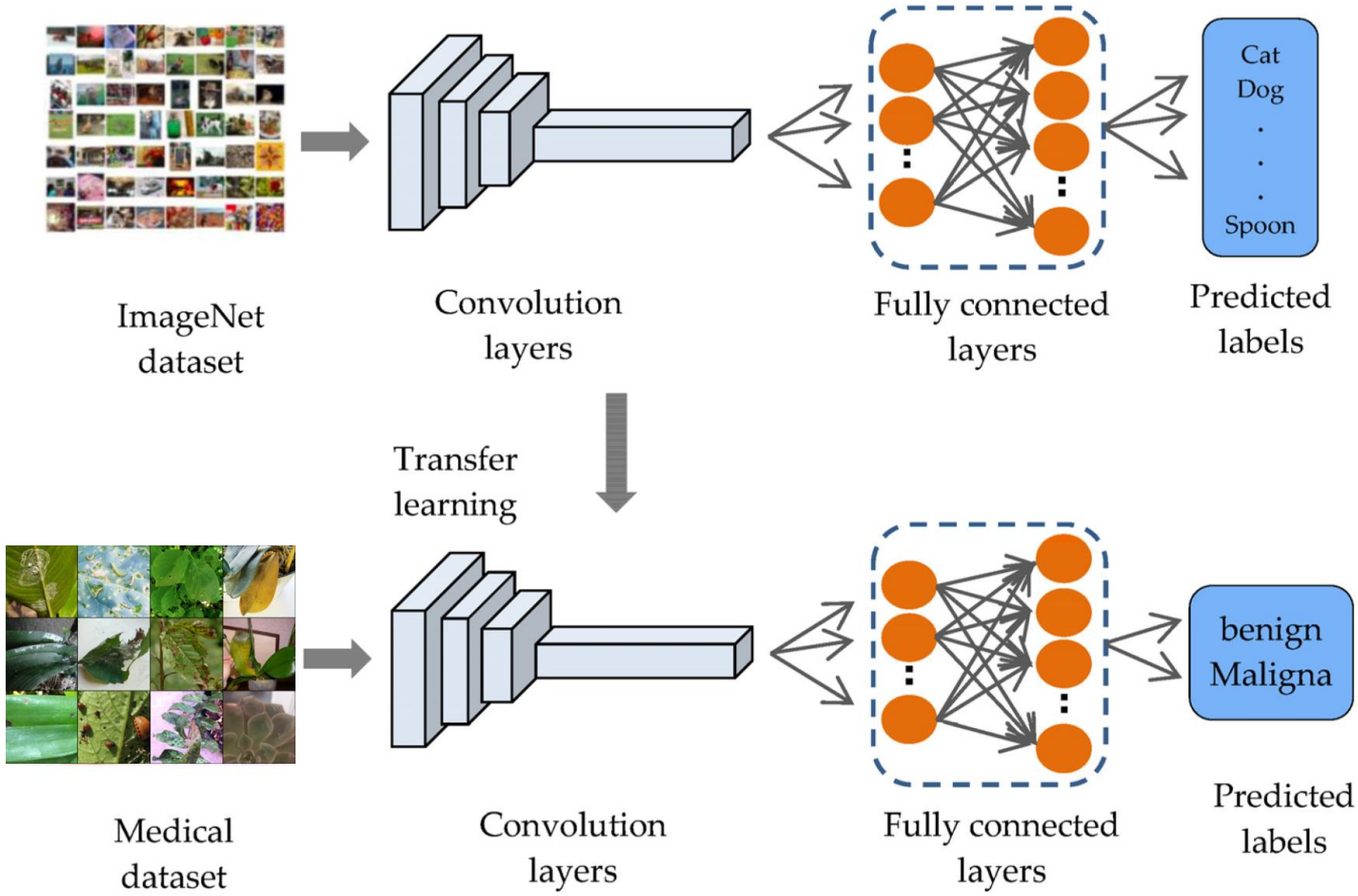
$$\text{F1} = \frac{2\times\text{Precision}\times\text{Recall}}{\text{Precision}+\text{Recall}}$$

Stages

1. Transfer learning

2. Retraining of the backbone networks with special loss functions (3 Siamese + 3 Cosine-based) + classification part

Transfer learning



Contrastive loss

Contrastive loss is one of the first loss functions used in Siamese networks. The loss takes as input embeddings of pairs of samples that are either similar or dissimilar and it brings similar samples closer and dissimilar samples farther apart. It can be formulated as in the equation:

$$\text{Loss} = (1 - y) \cdot \tfrac{1}{2} D^2 + y \cdot \tfrac{1}{2} \max(0, m - D)^2$$

where the y value is 1 if the image pairs are of the same class, and 0 if the image pairs are of different classes. The D variable is the Euclidean distance between the embeddings. The max function takes the larger value of 0 and the margin (typically this value is set to 1) minus the distance.

Triplet loss

The triplet loss function uses three images during evaluation. The anchor is an arbitrary data point. The positive image belongs to the same class as the anchor. The negative image belongs to a different class from the anchor. The triplet loss reduces the distance between the anchor and the positive image while increasing the distance between the anchor and the negative image. It can be formulated as shown in the equation:

$$L_{triplet} = \max(D_{ap} - D_{an} + \text{margin}, 0)$$

Where D_{ap} is the square of the Euclidean distance between the embeddings of the anchor and the positive image, and D_{an} is the square of the Euclidean distance between the embeddings of the anchor and the negative image. The margin is the desired difference between the anchor-positive distance and the anchor-negative distance. The margin is set to 0.2 in the original paper, while in the current research, the margin is set to 1.

Quadruplet loss

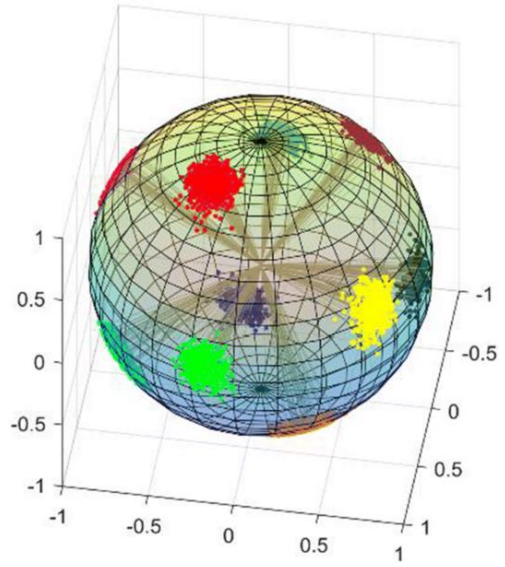
The quadruplet loss function is a generalization of triplet loss with a constraint involving four inputs: an anchor, a positive input, and two negative inputs. Quadruplet loss is designed to ensure a smaller intra-class variation and a larger inter-class variation in the embedding space, which leads to better performance in downstream tasks. It can be formulated as shown in the equation:

$$L_{quadruple} = \max(D_{ap} - D_{an1} + \text{margin1}, 0) + \max(D_{ap} - D_{an2} + \text{margin2}, 0)$$

Where D_{ap} is the square of the Euclidean distance between the embeddings of the anchor and the positive image. D_{an1} and D_{an2} are the squares of the Euclidean distances between the embeddings of the anchor and the first and second negative images, respectively. The margins margin1margin1 and margin2margin2 are the desired differences between the pairs' distances. The margins are set to 1 and 0.5 in the original paper, while in the current research, the margins are set to 2 and 1.

Alternatives to Siamese networks - loss functions based on angular spans

To organize the training process for the aforementioned loss functions, data preparation is essential. The selection of pairs, triplets, and quadruplets of data samples offers advantages in terms of the variety of combinations, but it also comes with drawbacks related to resource consumption.



Loss functions based on angular spans provide an alternative approach to Siamese networks. While Siamese networks operate in Euclidean space, angular spans functions work in angular space. Despite this distinction, their objective remains the same: to minimize variations within a class and maximize variations between classes of image embeddings.

The SphereFace (A-Softmax) loss function imposes discriminative constraints on a hypersphere manifold, which intrinsically matches the prior assumption that images also lie on a manifold. Moreover, the size of the angular margin can be quantitatively adjusted by a special parameter. The A-Softmax loss function is defined as in Equation (4):

$$L_{\text{ang}} = \frac{1}{N}\sum_{i} -\log\left(\frac{e^{\|\boldsymbol{x}_i\|\psi(\theta_{y_i,i})}}{e^{\|\boldsymbol{x}_i\|\psi(\theta_{y_i,i})} + \sum_{j\neq y_i} e^{\|\boldsymbol{x}_i\|\cos(\theta_{j,i})}}\right)$$

Where N is the total number of training samples. xi and yi denote the input feature vector and the class label for the i-th training sample, respectively. The function $\psi(\theta_{yi,i})=(-1)^k\cos(m\theta_{yi,i})-2k$ where $\theta_{yi,i}\in[k\pi/m,(k+1)\pi/m)$ and $k\in[0, m-1]$. Here, $m \geq 1$ is an integer that controls the size of the angular margin. In the original paper and in current research, the margin m is set to 4.

CosFace and ArcFace loss

The CosFace (large margin cosine) loss function [51] reformulates the softmax loss as a cosine loss by L2-normalizing both features and weight vectors to remove radial variations. Based on this normalization, a cosine margin term is introduced to further maximize the decision margin in the angular space. As a result, minimum intra-class variance and maximum inter-class variance are achieved through normalization and cosine decision margin maximization. It can be formulated as in Equation:

$$L = -\frac{1}{N}\sum_{i=1}^{N}\log\frac{e^{s(\cos(\theta_{y_i,i})-m)}}{e^{s(\cos(\theta_{y_i,i})-m)}+\sum_{j\neq y_i}e^{s\cos(\theta_j)}}$$

Where N is the number of training samples, xi is the i-th feature vector corresponding to the ground-truth class of yi, the Wj is the weight vector of the j-th class. The θj is the angle between Wj and xi. M is the margin which controls the cosine margin. S is the scale parameter, used to scale the logits for numerical stability and gradient control. In the original paper, the margin m was varied between 0.25 and 0.45, while the scale s was set to 64. However, in current research settings, the margin mmm is fixed at 0.4 and the scale s is set to 32.

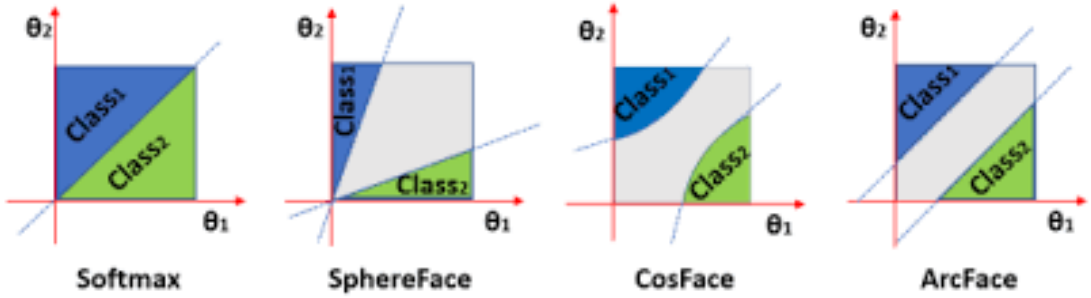
CosFace and ArcFace loss

The ArcFace (Additive Angular Margin) loss function [51] incorporates the Additive Angular Margin Loss, which directly optimizes the angular margin between different classes. It can be formulated as follows:

$$L_3 = -\frac{1}{N}\sum_{i=1}^{N}\log\frac{e^{s(\cos(\theta_{y_i}+m))}}{e^{s(\cos(\theta_{y_i}+m))}+\sum_{j=1,j\neq y_i}^{n}e^{s\cos\theta_j}}$$

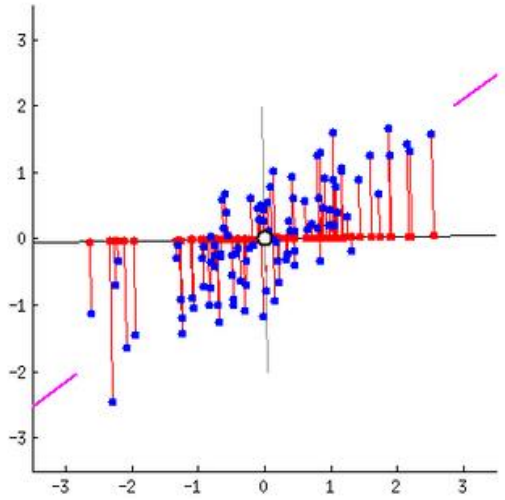
Where N is the number of training samples, θj is the angle between the weight Wj and the feature xi. M is a margin and s is a scale. The original paper used 0.5 radians (28.6 degrees) for m and 64 for s. In the current research margins is set to 0.5 and scale is set to 32.

Comparing the classification boundary under binary classification case. ArcFace has a constant linear angular margin throughout the whole interval. By contrast, SphereFace and CosFace only have a nonlinear angular margin.

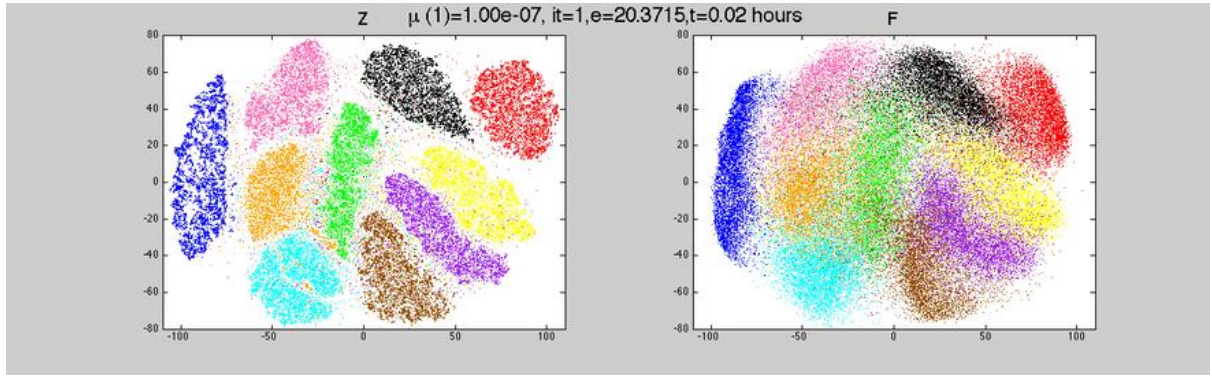


Embedding visualization

There are several methods of embedding relation visualization. T-SNE converts similarities between data points to joint probabilities and tries to minimize the Kullback-Leibler divergence between the joint probabilities of the low-dimensional embedding and the high-dimensional data. t-SNE has a cost function that is not convex, i.e. with different initializations we can get different results. Uniform Manifold Approximation and Projection (UMAP) is a dimension reduction technique that can be used for visualisation similarly to t-SNE, but also for general non-linear dimension reduction.
Principal Component Analysis (PCA) reduces the number of dimensions in large datasets to principal components that retain most of the original information. It does this by transforming potentially correlated variables into a smaller set of variables, called principal components. Besides using PCA as a data preparation technique, we can also use it to help visualize data. The PCA results is always reproducible. 2D and 3D plots were also used to estimate models embedding extraction abilities.



PCA



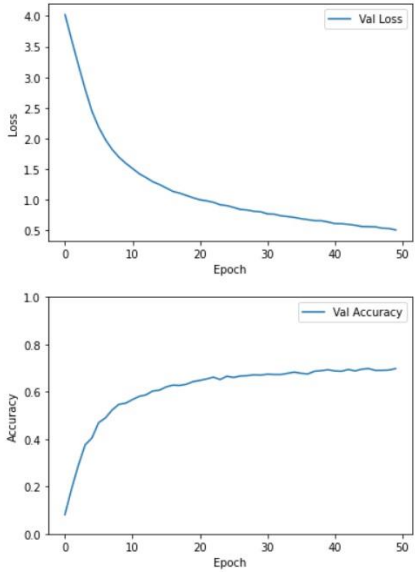
t-SNE

Infrastructure

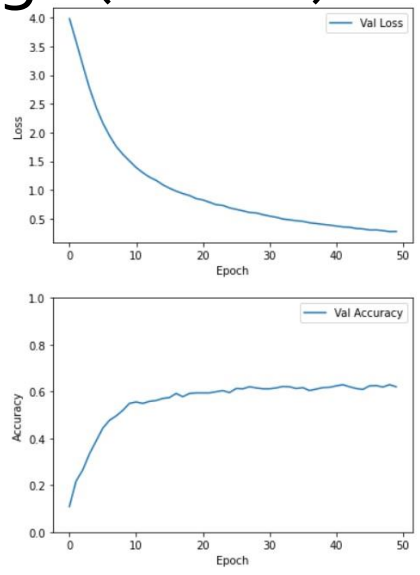
hYBRI | LIT/JINR

The heterogeneous infrastructure of the Joint Institute for Nuclear Research was used for the experiments. Models were trained on NVIDIA Volta V100 GPUs with 512 GB RAM boards.

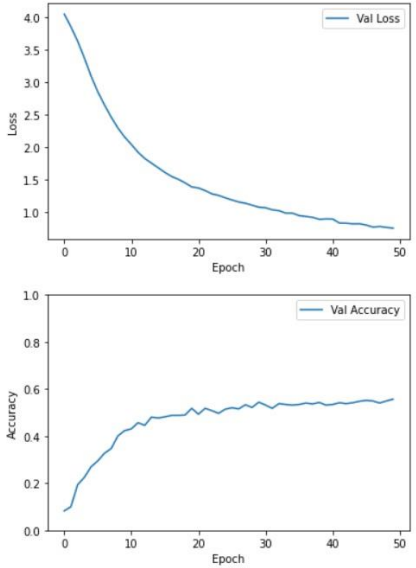
Results – first stage (curves)



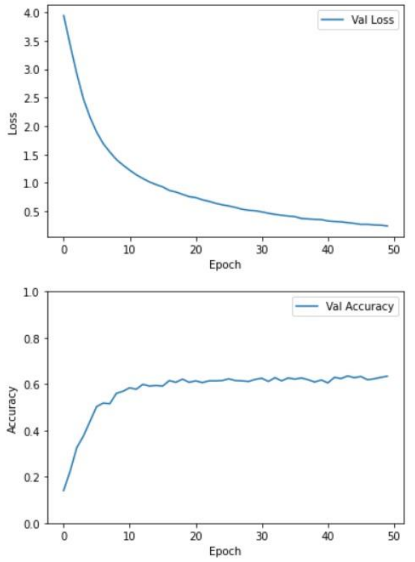
ConvNeXt_Small



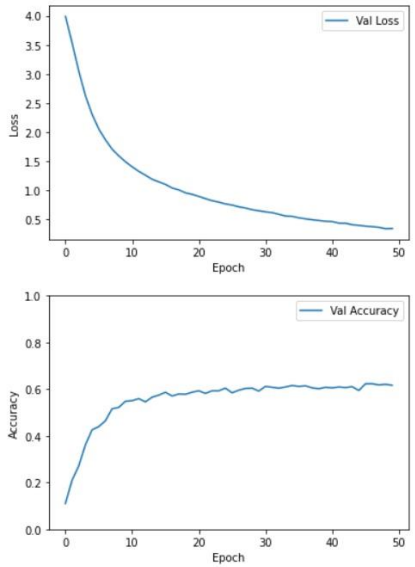
EfficientNet_B3



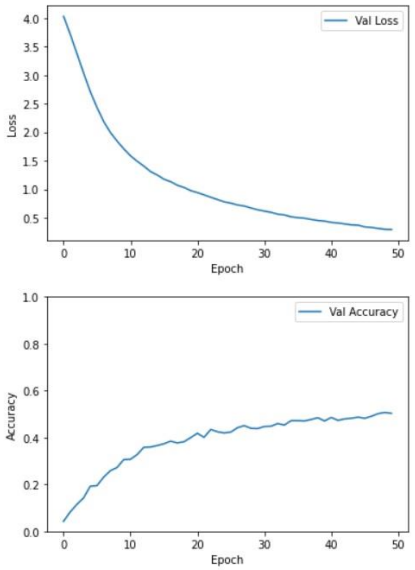
Inception_V3



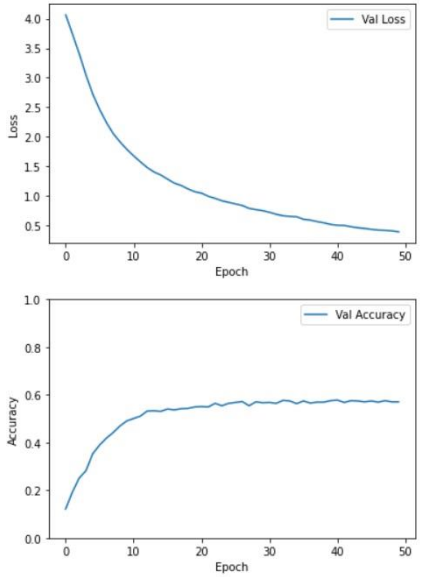
MobileNetV2



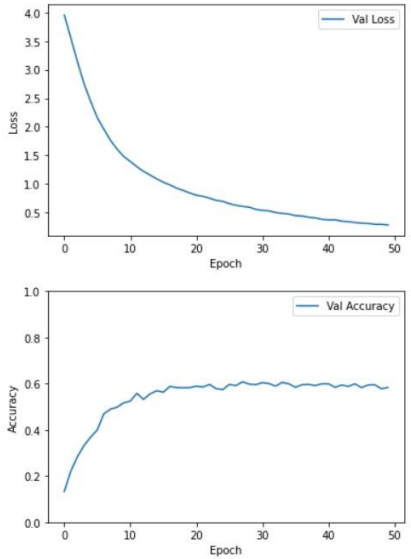
MobileNe_V3_Large



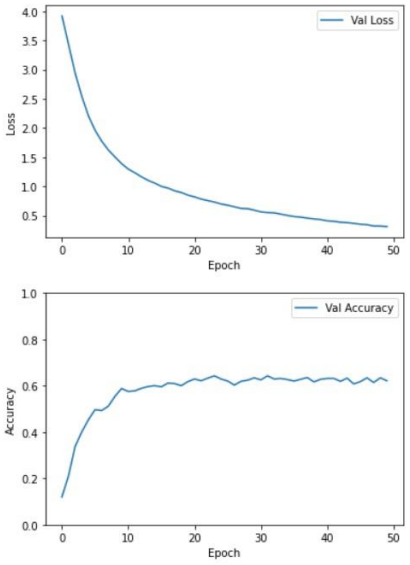
MNASNET



regnet_x_1_6gf



ResNet101



ResNeXt50_32x4d

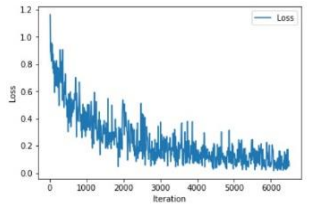
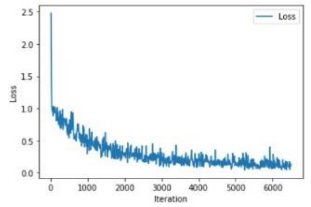
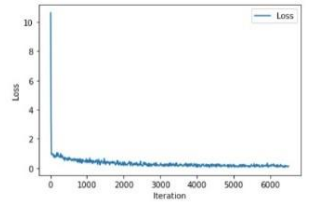
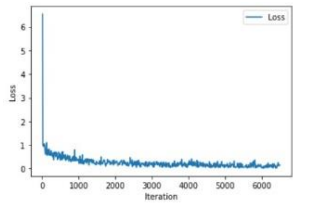
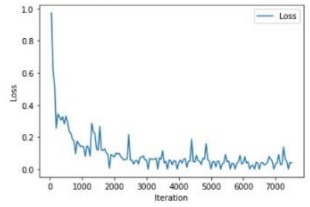
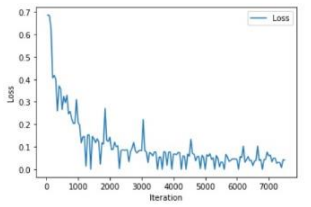
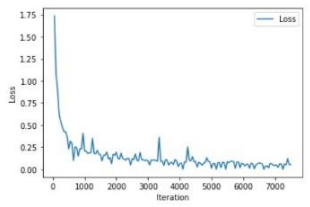
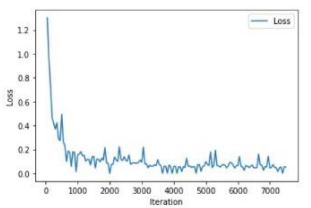
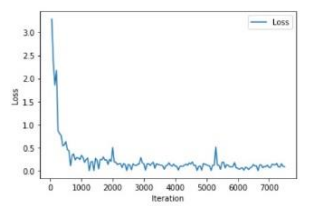
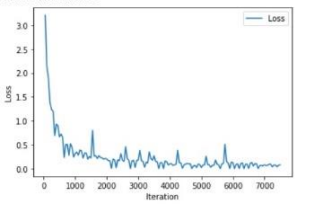
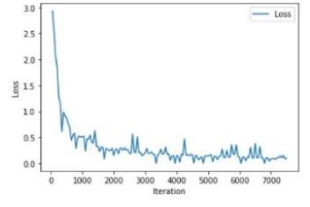
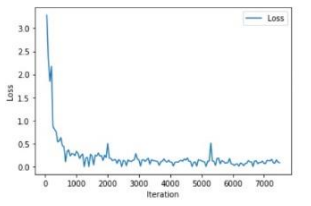
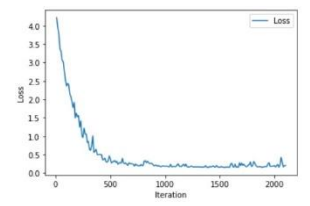
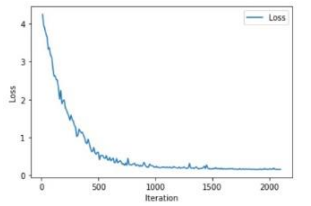
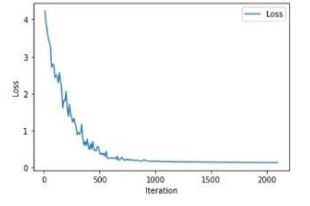
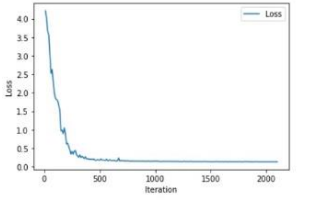
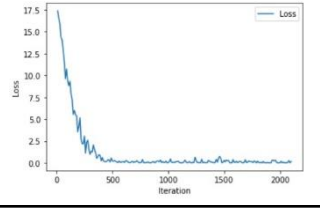
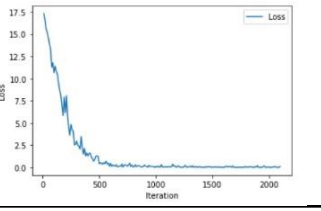
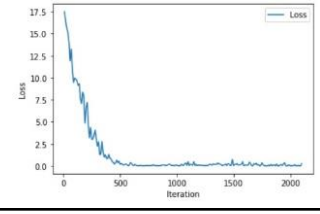
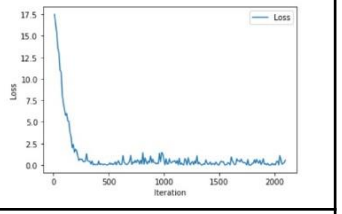
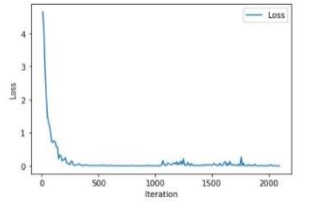
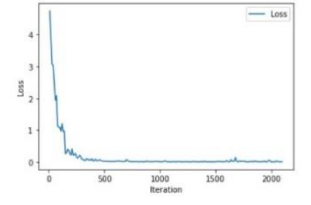
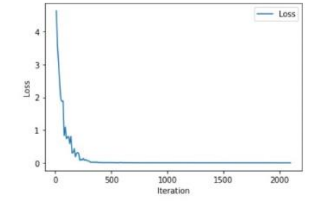
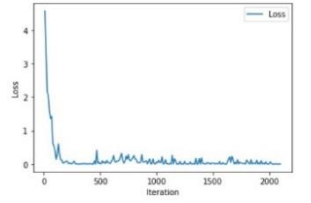
Results – first stage (metrics)

At the first stage, the weights of the selected backbone networks were frozen. The classification part of the networks was changed to two linear layers joined with a ReLU activation function. The CrossEntropy loss function and Adam optimizer with a learning rate of 0.0001 were used for training the networks for 50 epochs with a batch size of 64. Evaluation parameters obtained after 5 runs are presented in Table

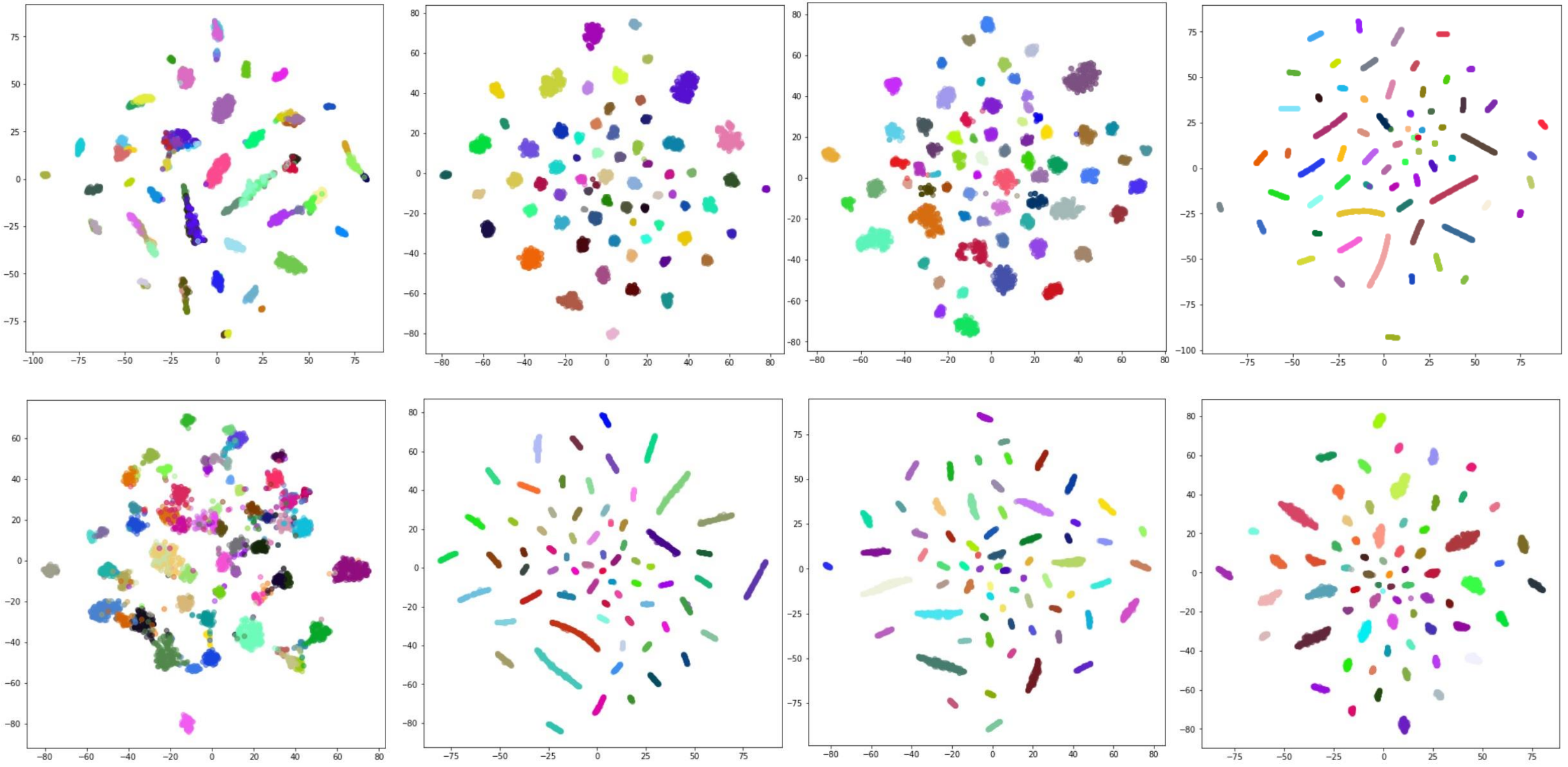
	Accuracy	MEPT	wa precision	wa recal	wa f1 score
ConvNeXt_small	70.41	17.84	0.71	0.70	0.69
EfficientNet_B3	62.87	5.36	0.64	0.62	0.61
Inception_V3	55.68	6.86	0.58	0.56	0.54
MNASNet1_3	54,09	3.25	0.56	0.54	0.53
MobileNetV2	62.79	2.81	0.64	0.63	0.62
MobileNet_V3_Large	61.87	2.48	0.62	0.62	0.61
RegNet_X_1_6GF	62.25	4,78	0.63	0.61	0.61
ResNet101	62,62	9.02	0.65	0.64	0.62
ResNeXt50_32X4D	62,73	8.02	0.66	0.64	0.63

Due to limitation in training resources and some rational consideration numbers of models have been excluded from the further experiments. The Inception_V3 and MNASNet1_3 was excluded due too low accuracy. MobileNet_V3_Large show worthier accuracy than MobileNetV2 so only las one were keeped. ConvNeXt shows best result on accuracy but MEPT of the model is very big. In further stages training time will also increase especially while using Siamese-approache still it is very promising results and ConvNeXt keepet for second stage. ResNeXt50_32X4D shows slightly better results than ResNet101 so it is passed to the second stage. EfficientNet_B3 also show good results so for models - two large and two small we selected for stage 2 experiment.

Results – second stage embedding extraction

	ConvNeXt_small	EfficientNet_B3	MobileNetV2	ResNeXt50_32X4D
Contrastive 50 epoch batch 32				
Triplet 50 epoch batch 32				
Quadruplet 50 epoch batch 32				
SphereFace 30 epoch batch 64				
CosFace 30 epoch batch 64				
ArcFace 30 epoch batch 64				

t-SNE

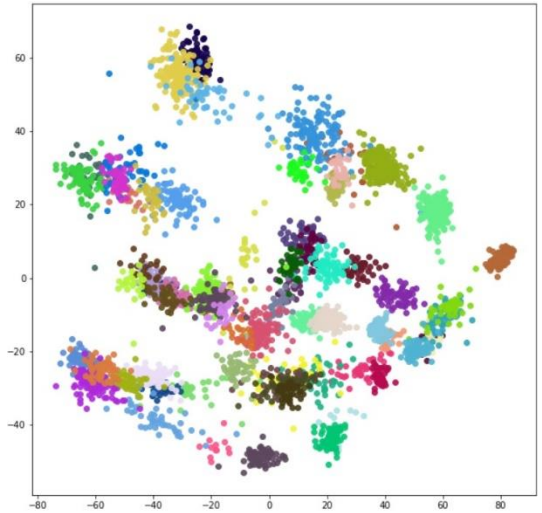


Embedding visualization PCA

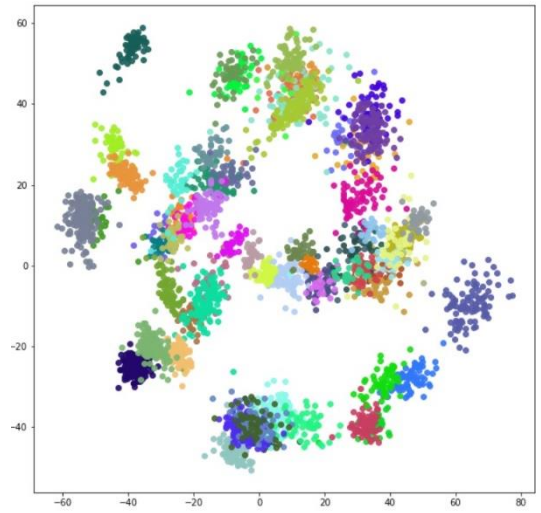
Normalization effect

Embedding visualization PCA (Contrastive loss normalization)

ConvNeXt_small

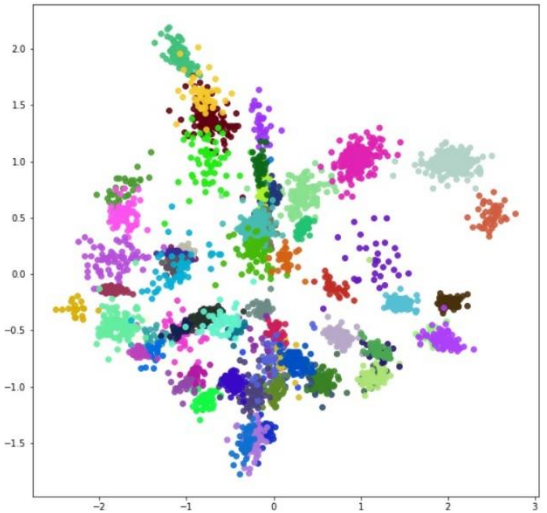


vanilla

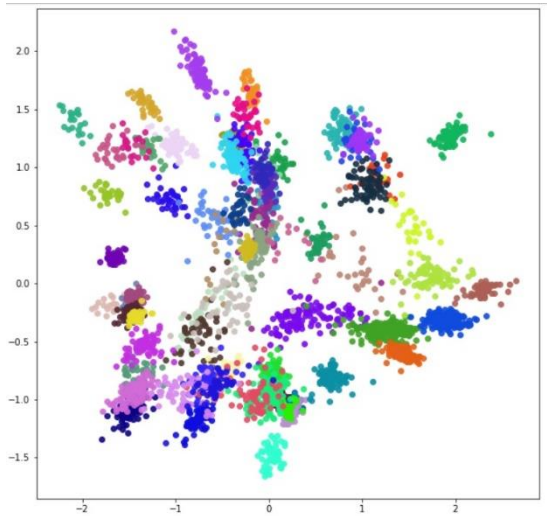


normalized

EfficientNet_B3

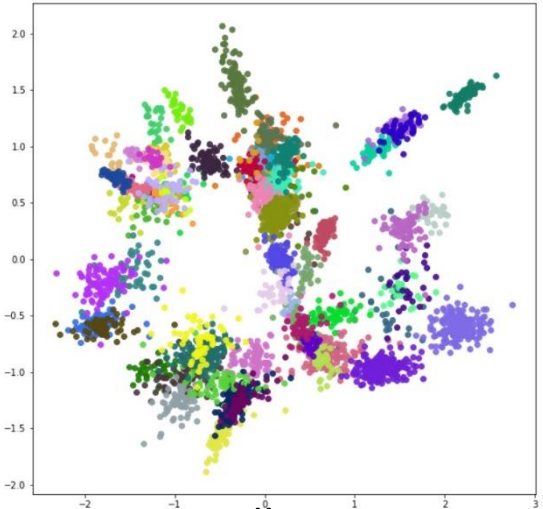


vanilla

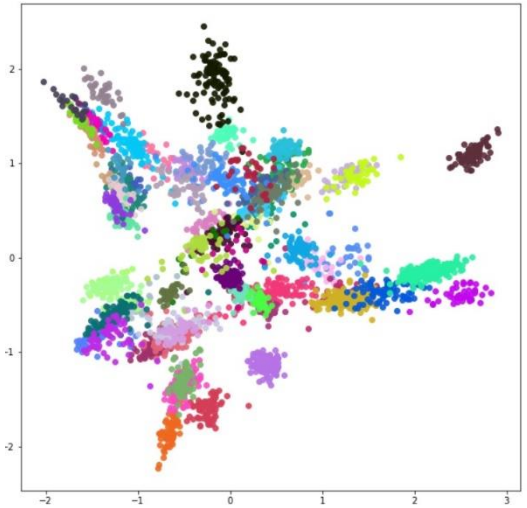


normalized

MobileNetV2

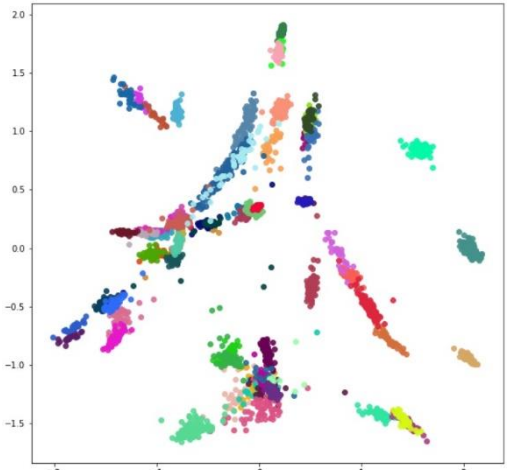


vanilla

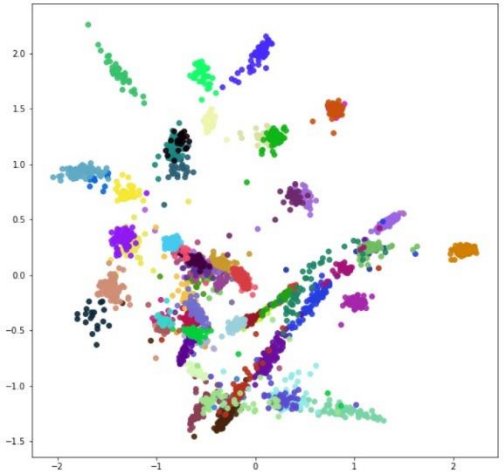


normalized

ResNeXt50_32x4d



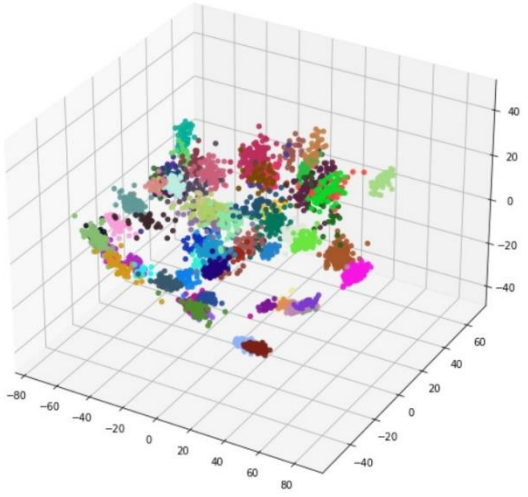
vanilla



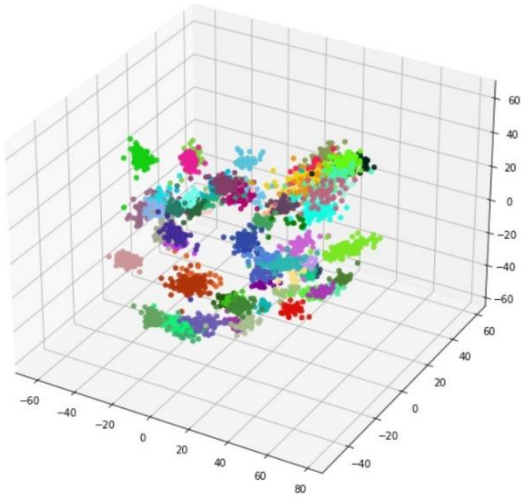
normalized

Embedding visualization3D PCA (Contrastive loss normalization)

ConvNeXt_small

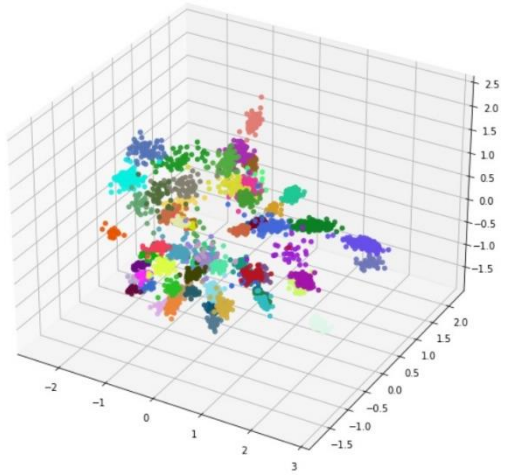


vanilla

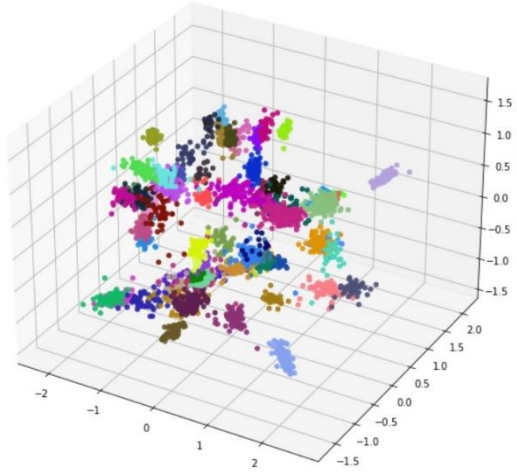


normalized

EfficientNet_B3

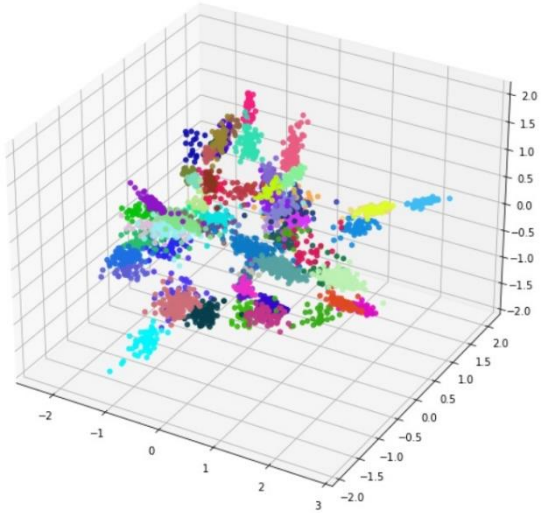


vanilla

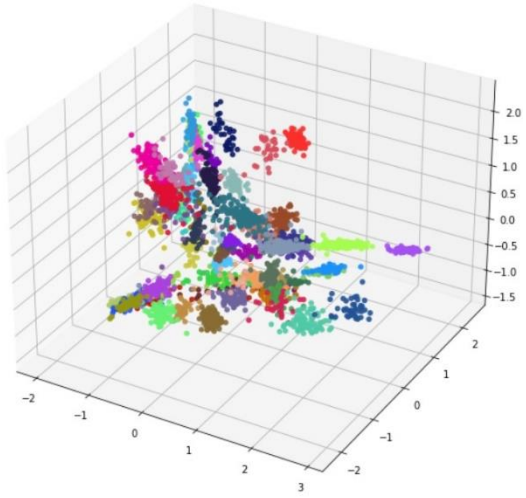


normalized

MobileNetV2

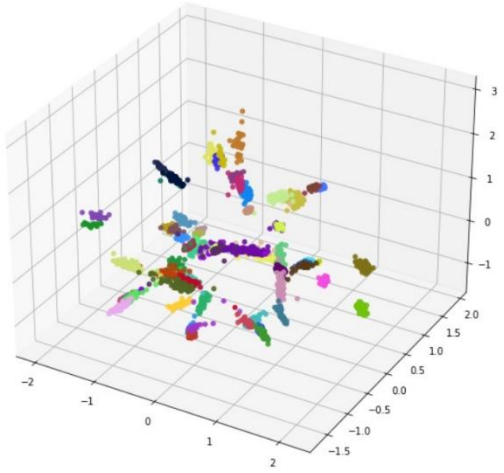


vanilla

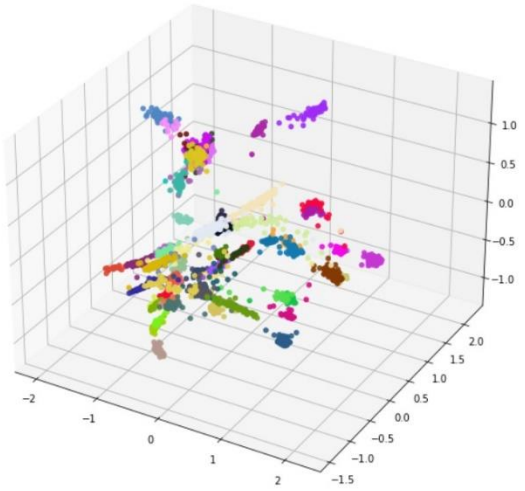


normalized

ResNeXt50_32x4d



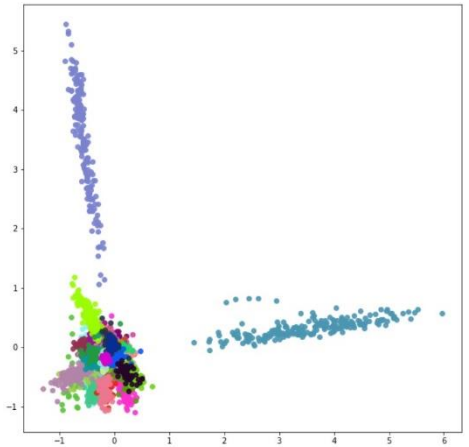
vanilla



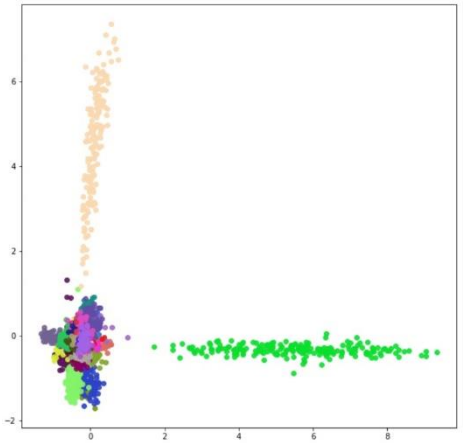
normalized

Embedding visualization (CosFace loss normalization)

ConvNeXt_small

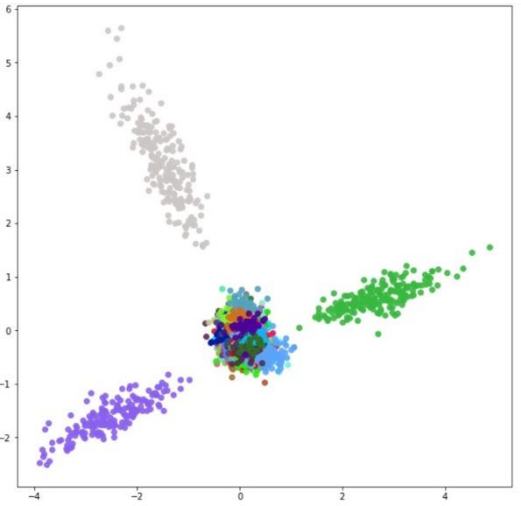


vanilla

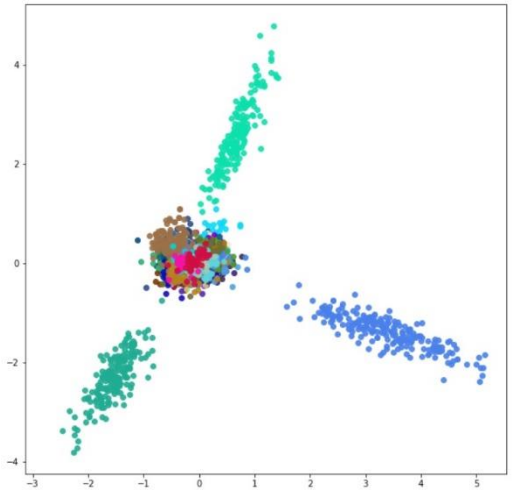


normalized

EfficientNet_B3

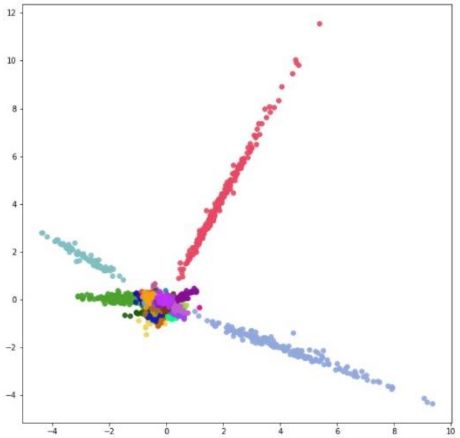


vanilla

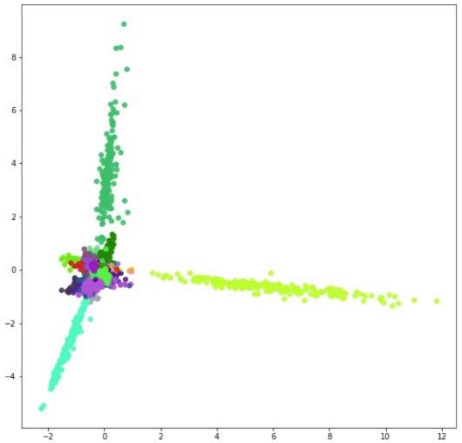


normalized

MobileNetV2

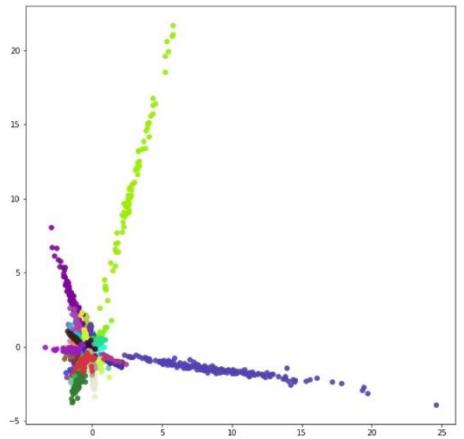


vanilla

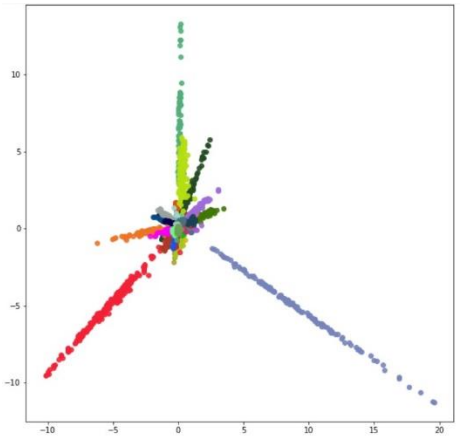


normalized

ResNeXt50_32x4d



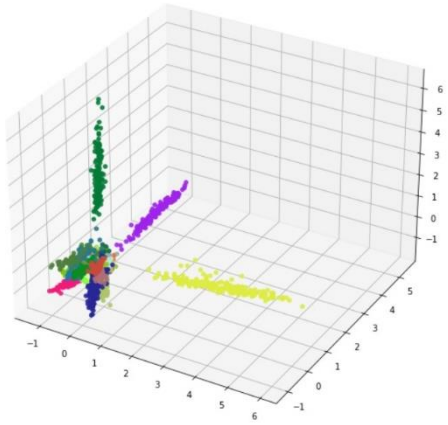
vanilla



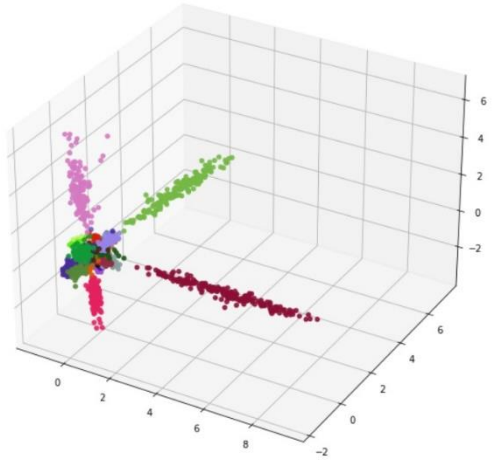
normalized

Embedding visualization 3D (CosFace loss normalization)

ConvNeXt_small

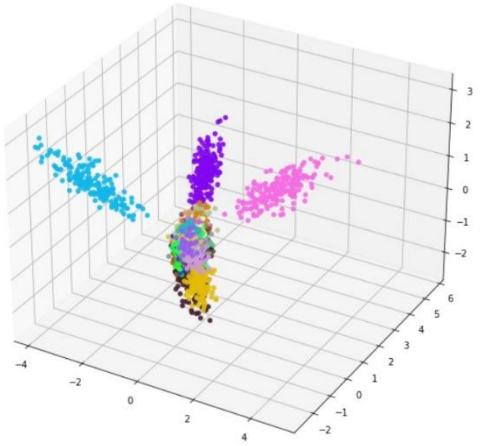


vanilla

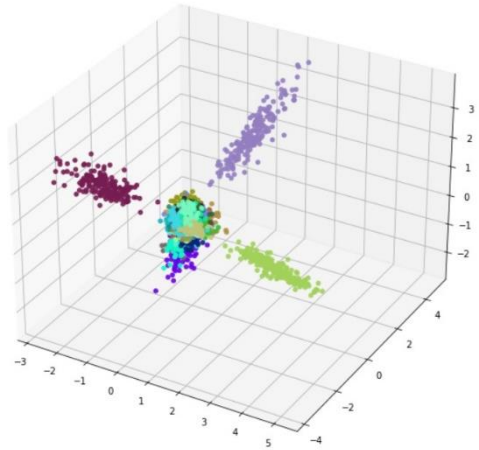


normalized

EfficientNet_B3

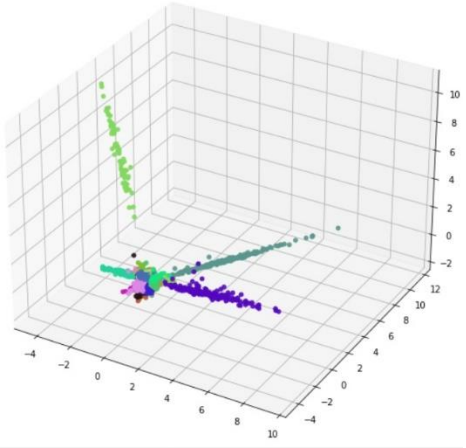


vanilla

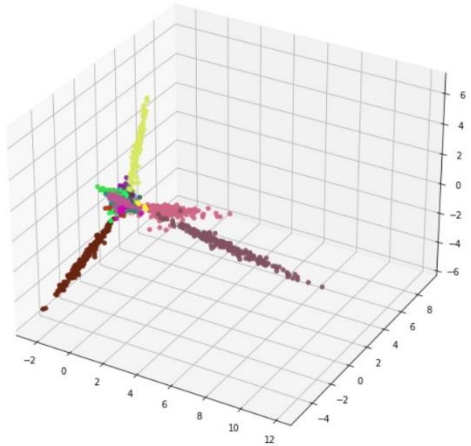


normalized

MobileNetV2

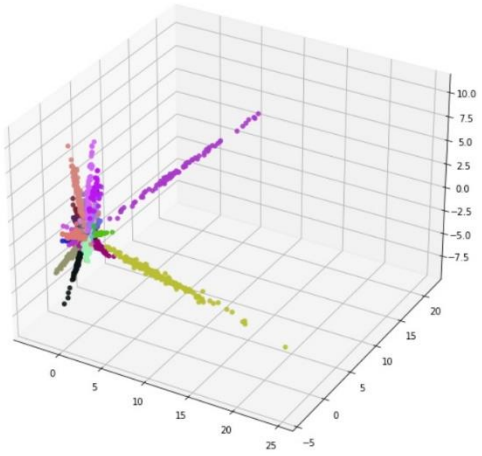


vanilla

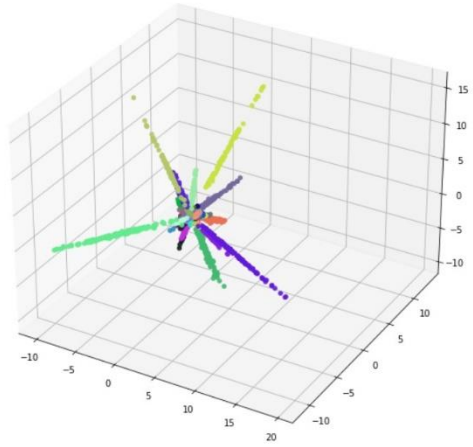


normalized

ResNeXt50_32x4d

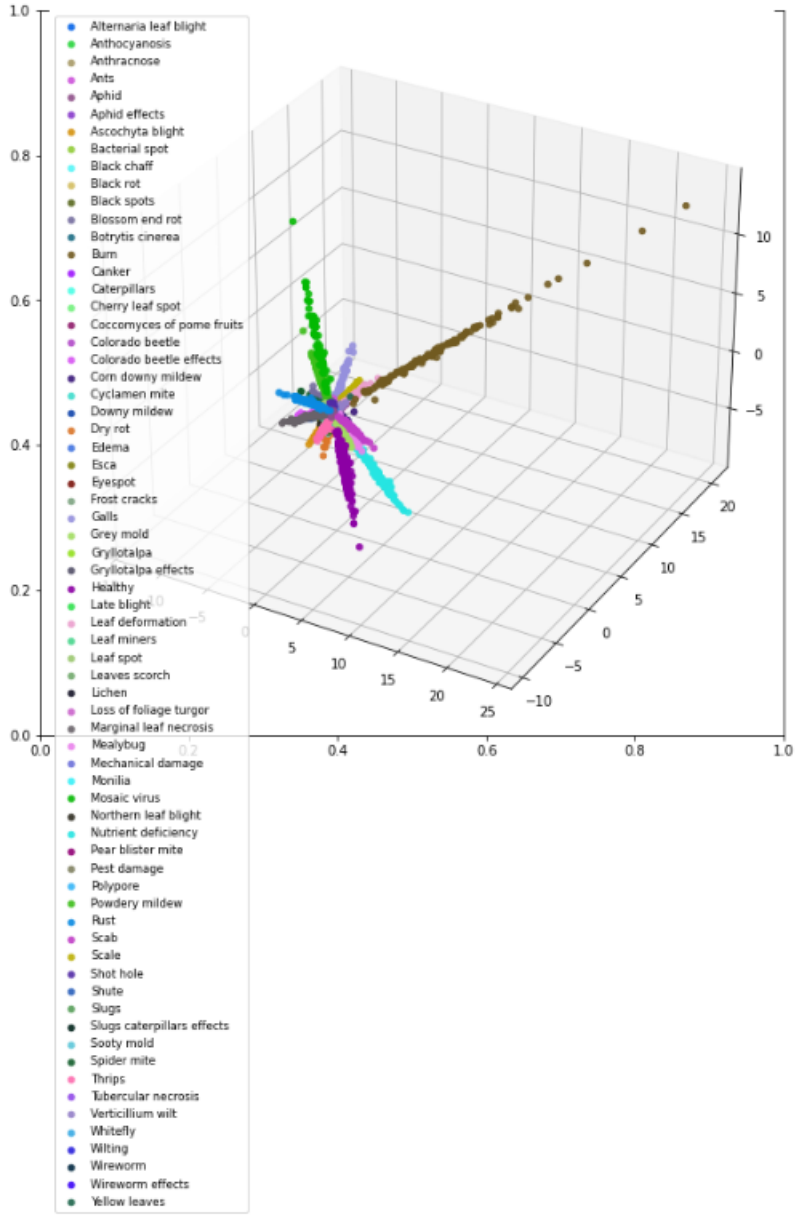


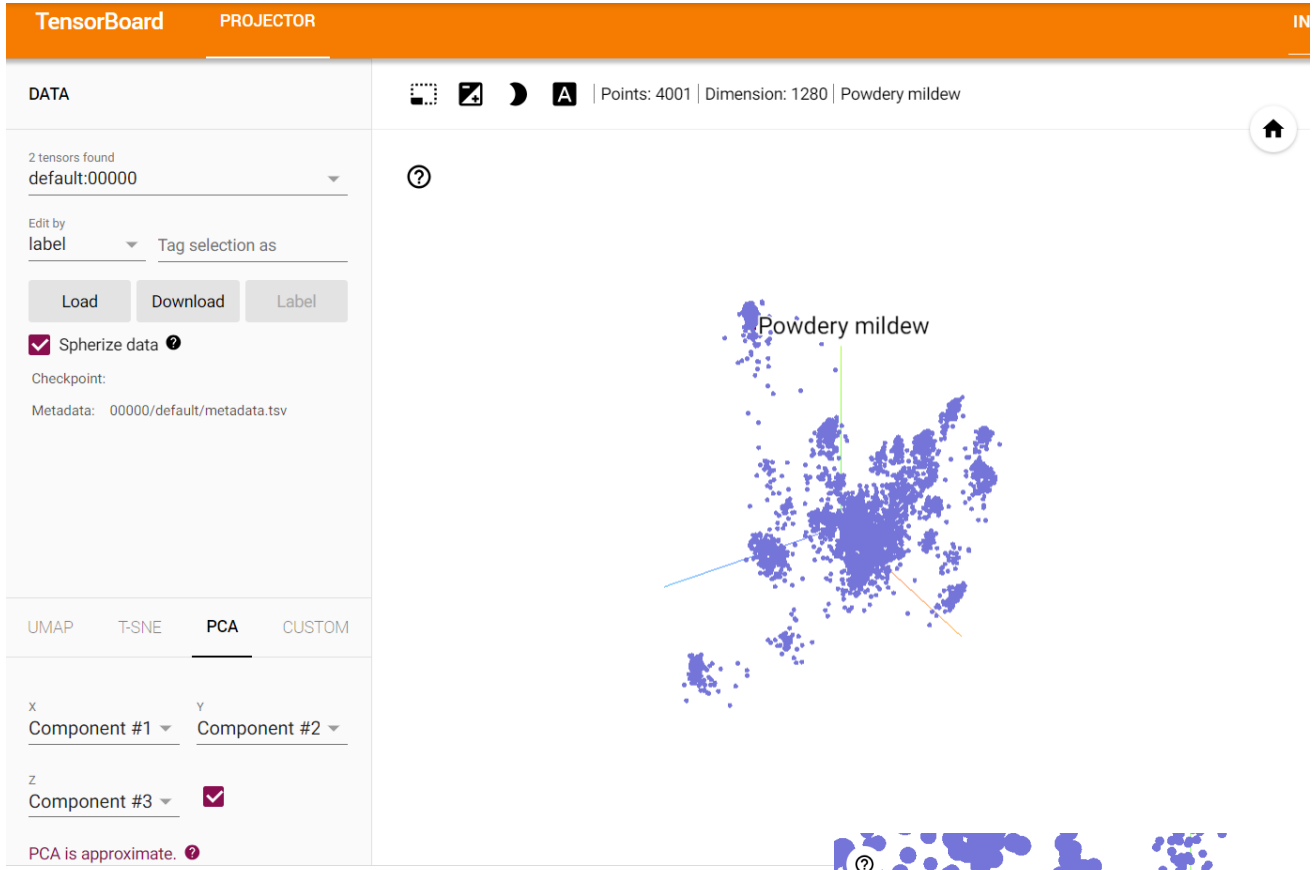
vanilla

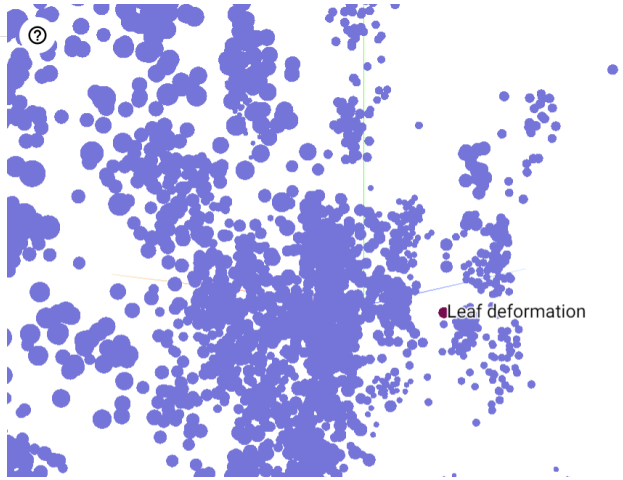


normalized

Tensorboard





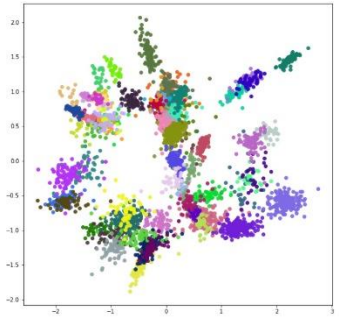


Embedding visualization PCA

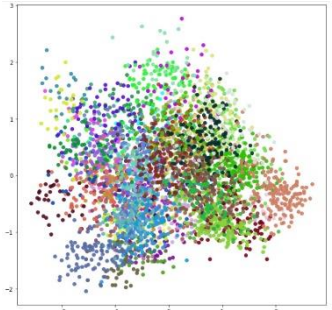
Similarity learning

MobileNetV2

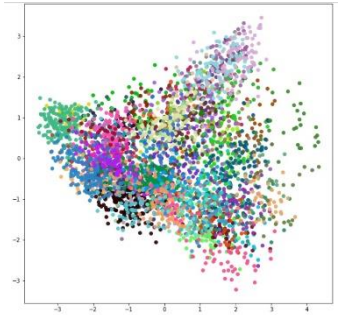
Contrastive



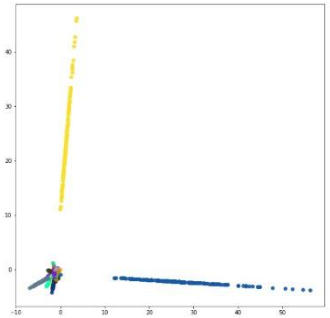
Triplet



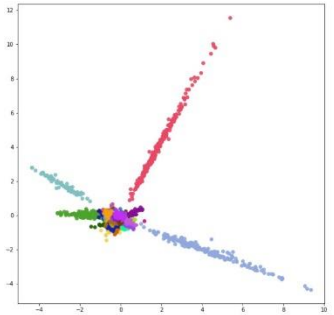
Quadruplet



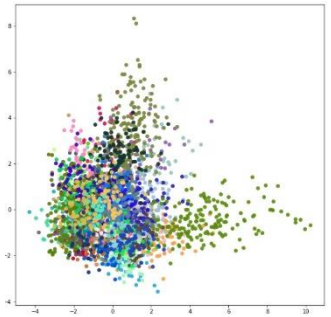
SphereFace



CosFace

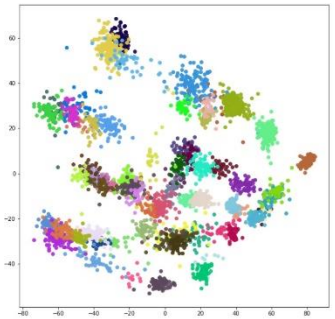


ArcFace

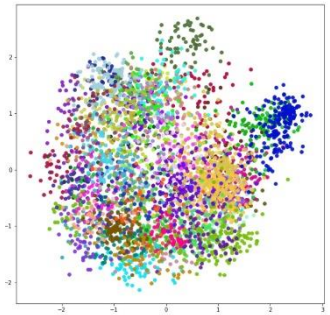


ConvNeXt_small

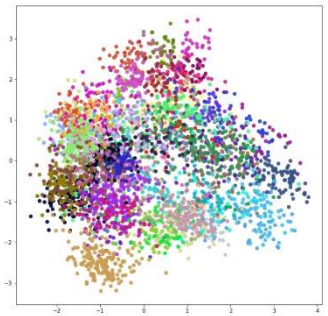
Contrastive



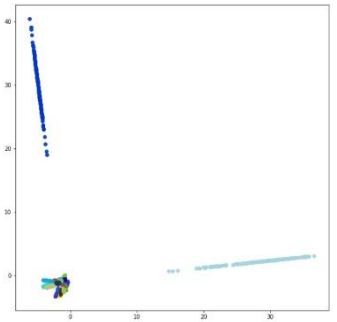
Triplet



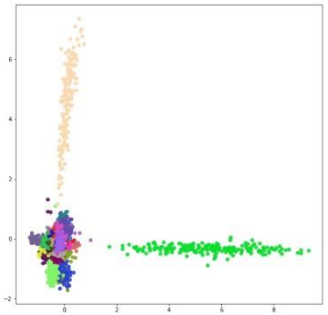
Quadruplet



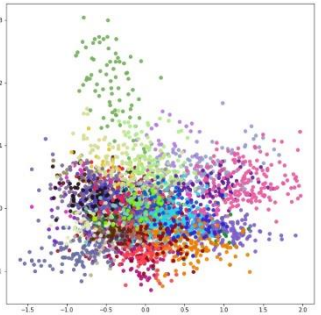
SphereFace



CosFace

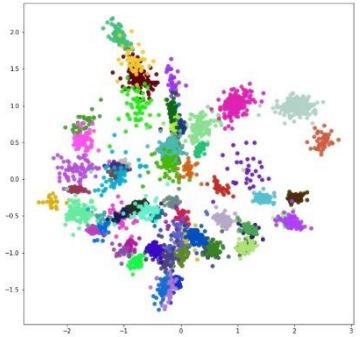


ArcFace

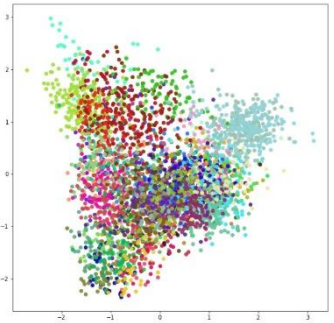


EfficientNet_B3

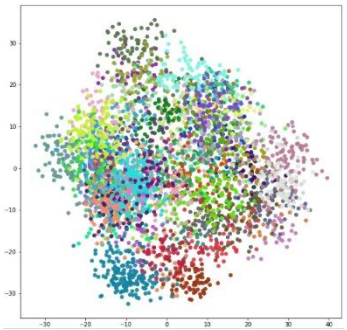
Contrastive



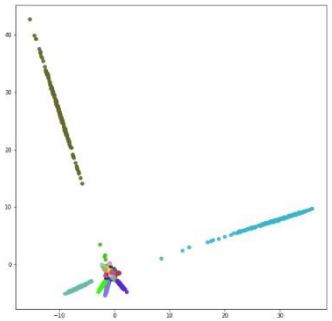
Triplet



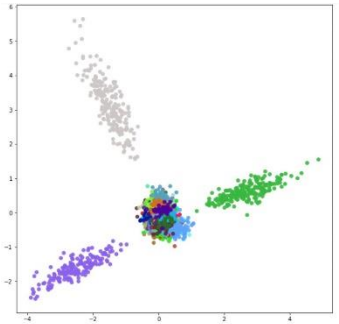
Quadruplet



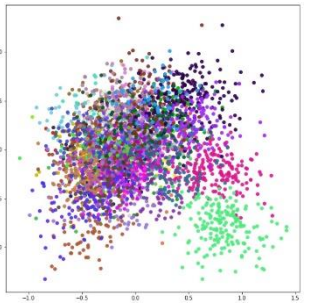
SphereFace



CosFace

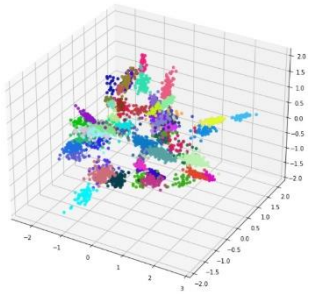


ArcFace

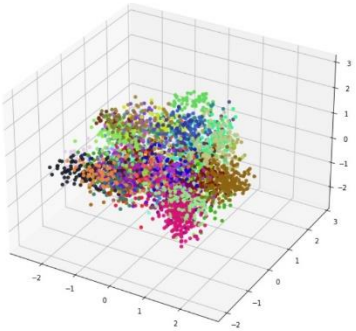


MobileNetV2

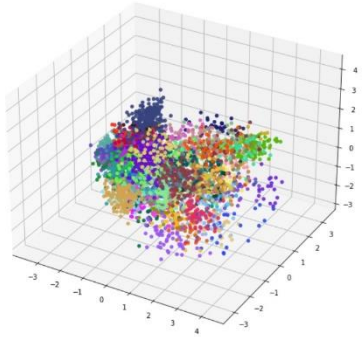
Contrastive



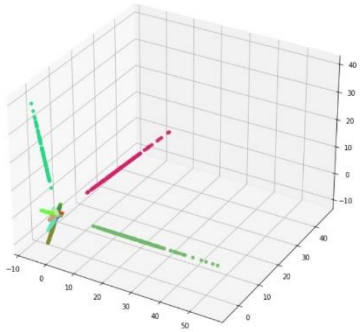
Triplet



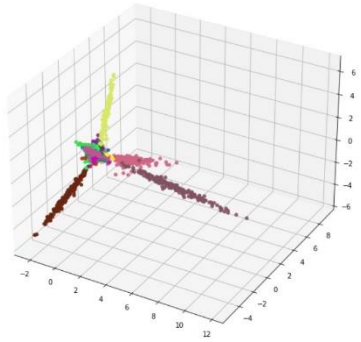
Quadruplet



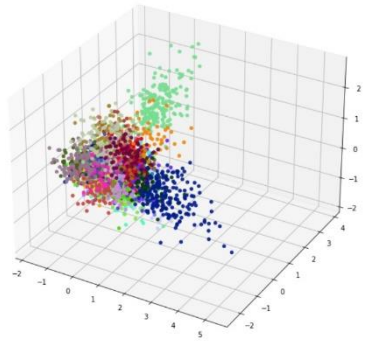
SphereFace



CosFace

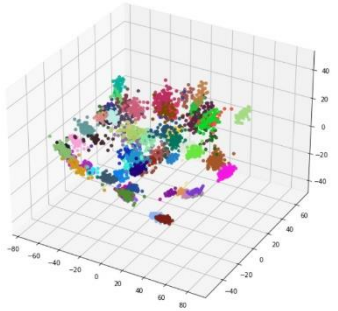


ArcFace

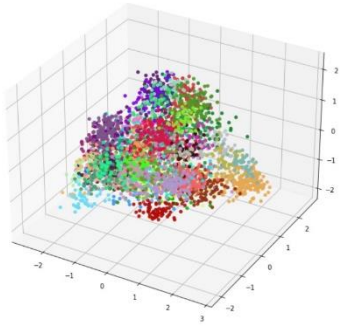


ConvNeXt_small

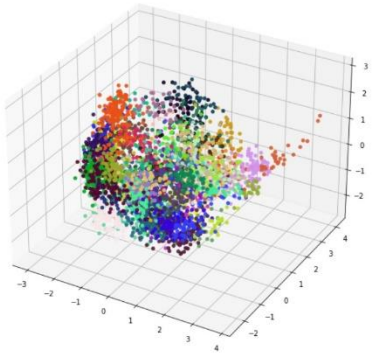
Contrastive



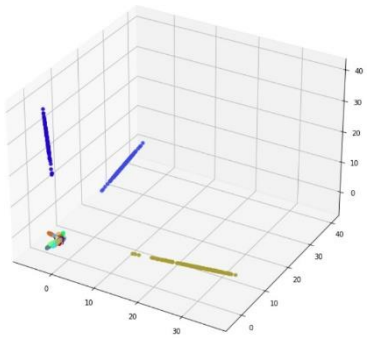
Triplet



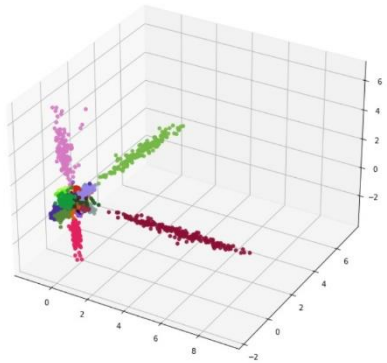
Quadruplet



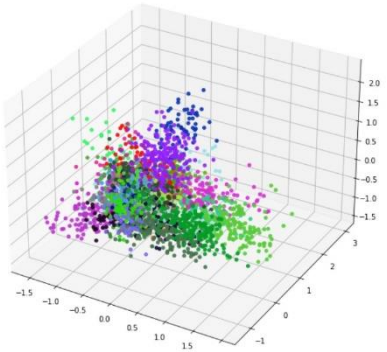
SphereFace



CosFace

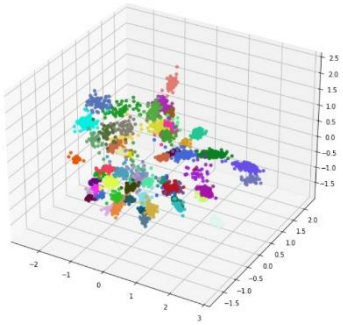


ArcFace

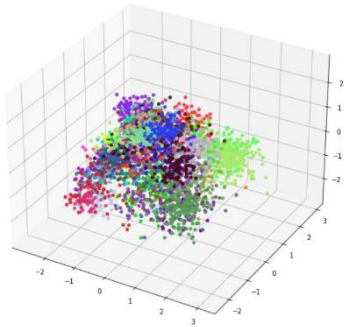


EfficientNet_B3

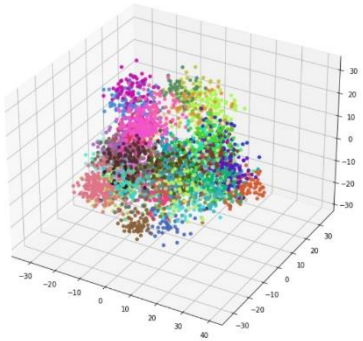
Contrastive



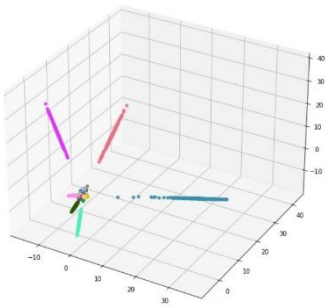
Triplet



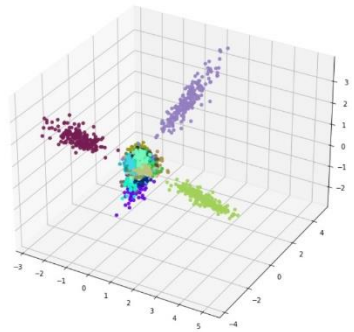
Quadruplet



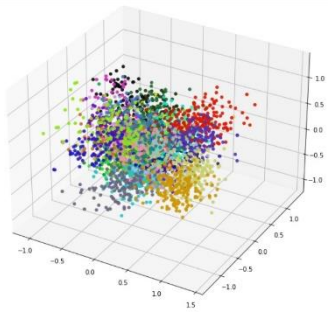
SphereFace



CosFace



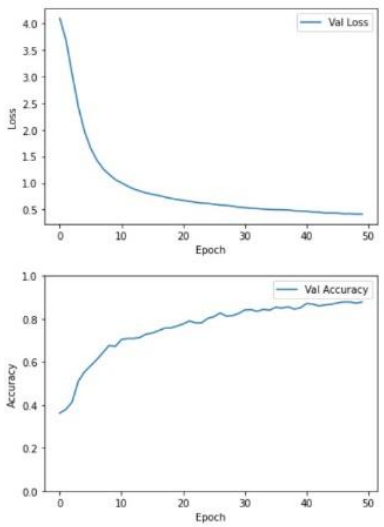
ArcFace



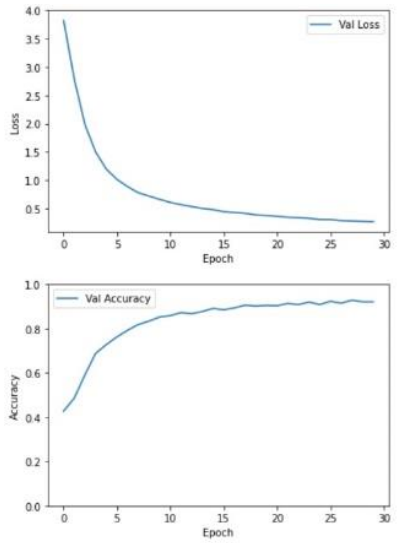
Classification

Results – second stage (Contrastive)

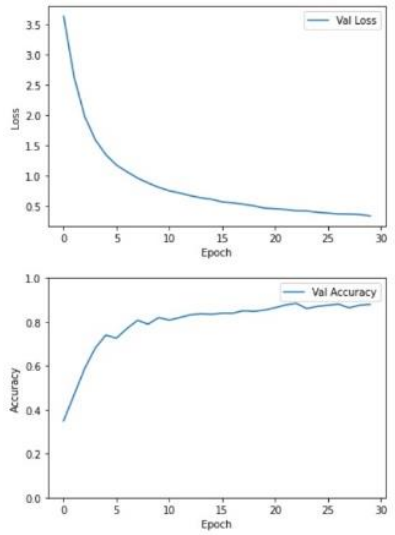
convnext_small



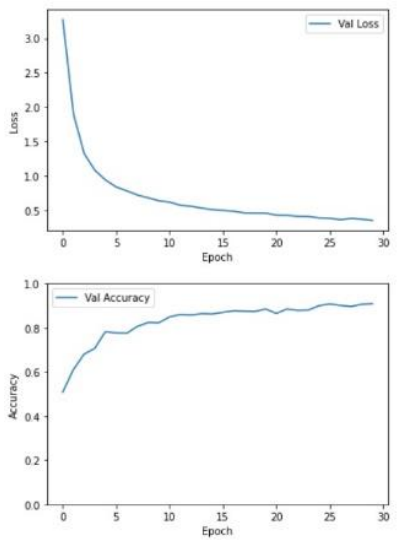
efficientnet_b3



mb_net_v2



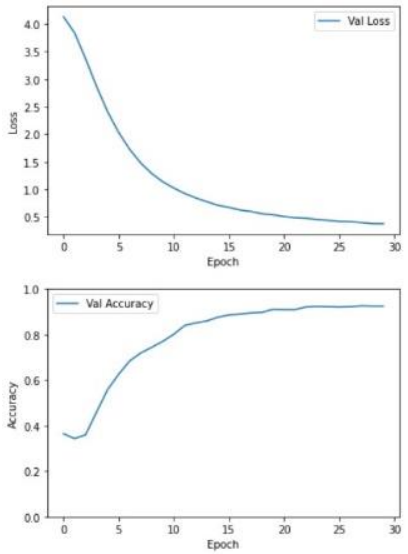
resnext50_32x4d



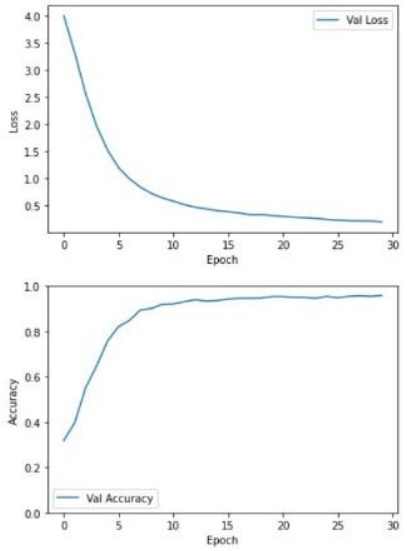
	Accuracy	MEPT	wa precision	wa recal	wa f1 score
ConvNeXt_small	87.89	150.51	0.87	0.86	0.85
ConvNeXt_small (norm)	88.03	148.43	0.89	0.88	0.86
EfficientNet_B3	93.62	45.54	0.94	0.94	0.94
EfficientNet_B3 (norm)	**95.15**	45.52	0.96	0.95	0.95
MobileNetV2	92.53	22.42	0.93	0.92	0.92
MobileNetV2 (norm)	90.94	22.37	0.92	0.91	0.90
ResNeXt50_32X4D	88,8	57.34	0.91	0.89	0.88
ResNeXt50_32X4D (norm)	84,64	54.63	0.91	0.91	0.90

Results – second stage (Triplet)

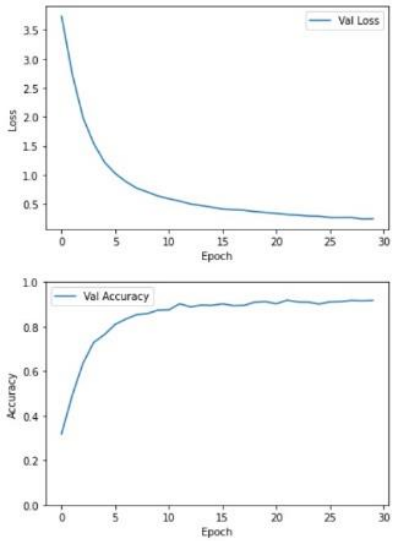
convnext_small



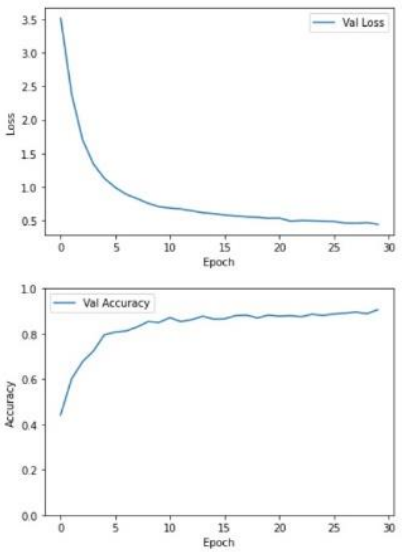
efficientnet_b3



mb_net_v2



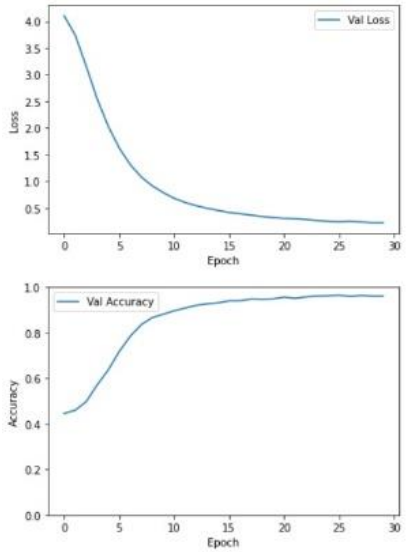
resnext50_32x4d



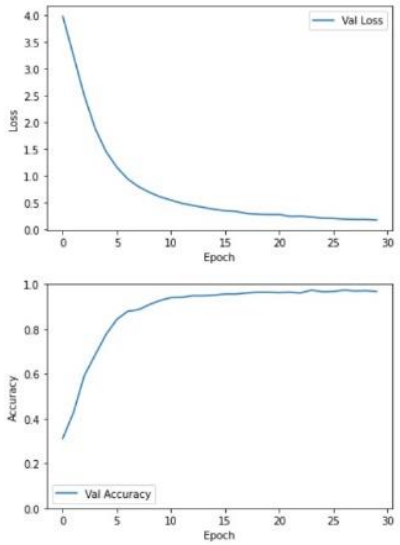
	Accuracy	MEPT	wa precision	wa recal	wa f1 score
ConvNeXt_small	92.88	213.62	0.94	0.94	0.93
ConvNeXt_small (norm)	92.56	214.31	0.93	0.93	0.93
EfficientNet_B3	95.33	67.13	0.95	0.95	0.95
EfficientNet_B3 (norm)	**95.73**	67.56	0.96	0.96	0.96
MobileNetV2	90.21	50.79	0.91	0.90	0.89
MobileNetV2 (norm)	90.89	52.93	0.92	0.92	0.92
ResNeXt50_32X4D	88,67	83.34	0.89	0.88	0.88
ResNeXt50_32X4D (norm)	89,56	82.63	0.89	0.89	0.88

Results – second stage (Quadruplet)

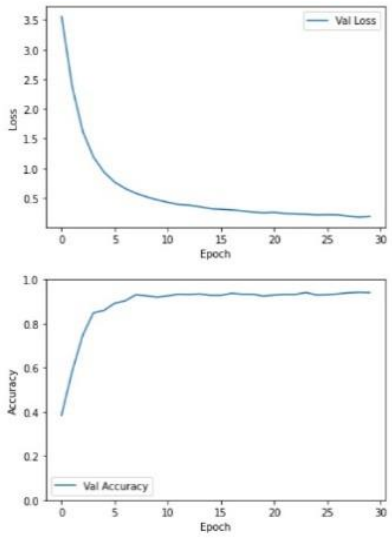
convnext_small



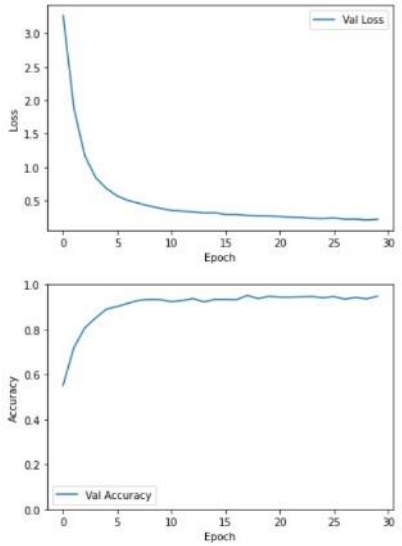
efficientnet_b3



mb_net_v2



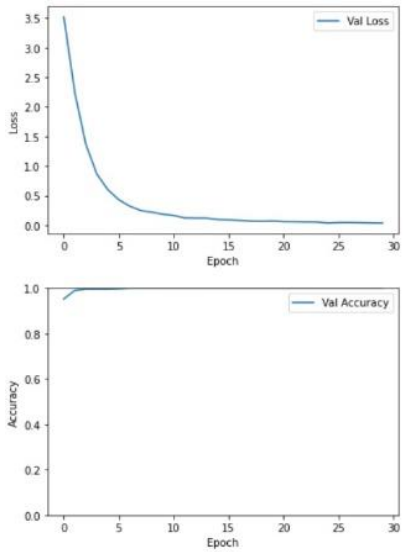
resnext50_32x4d



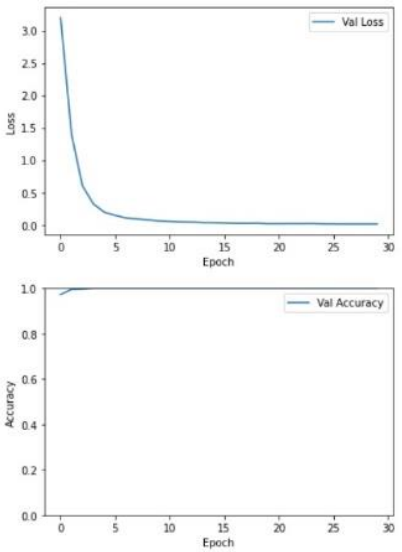
	Accuracy	MEPT	wa precision	wa recal	wa f1 score
ConvNeXt_small	96.12	278.98	0.97	0.97	0.96
ConvNeXt_small (norm)	96.75	279.41	0.97	0.97	0.97
EfficientNet_B3	**97.33**	86.85	0.98	0.98	0.98
EfficientNet_B3 (norm)	96.93	86.87	0.97	0.97	0.97
MobileNetV2	92.87	68.91	0.94	0.93	0.93
MobileNetV2 (norm)	93.79	68.65	0.95	0.94	0.94
ResNeXt50_32X4D	92.51	107.34	0.94	0.94	0.94
ResNeXt50_32X4D (norm)	93.79	107.16	0.95	0.94	0.94

Results – second stage (SphereFace)

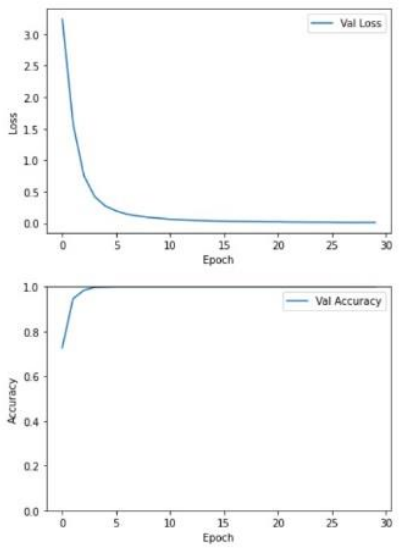
convnext_small



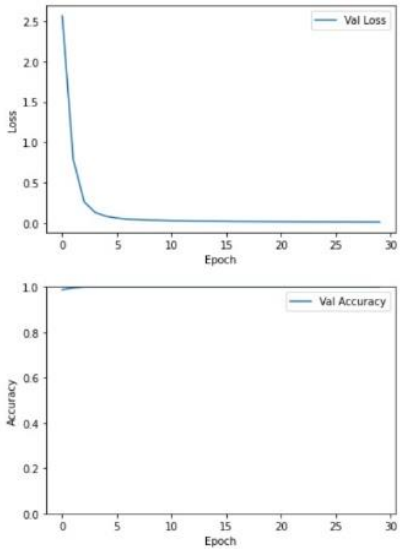
efficientnet_b3



mb_net_v2



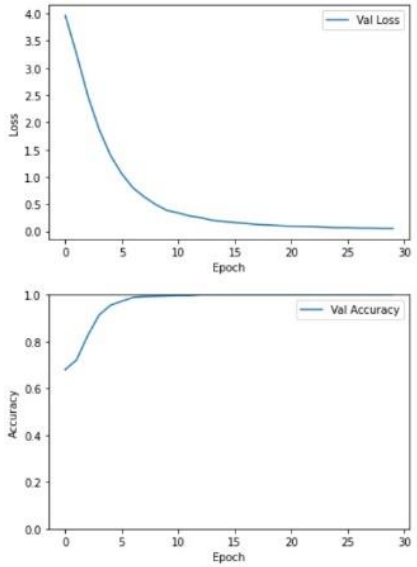
resnext50_32x4d



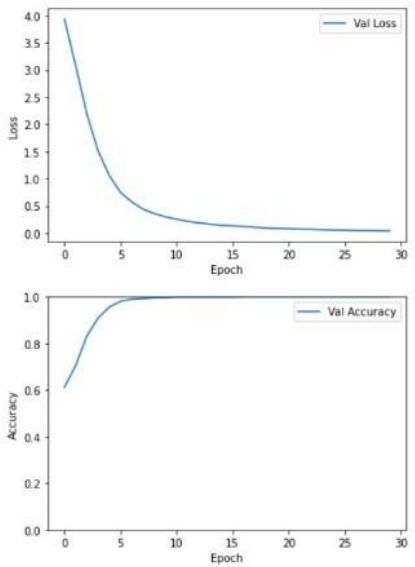
	Accuracy	MEPT	wa precision	wa recal	wa f1 score
ConvNeXt_small	100.0	86.05	1.00	1.00	1.00
ConvNeXt_small (norm)	99.87	86.17	1.00	1.00	1.00
EfficientNet_B3	99.95	26.03	1.00	1.00	1.00
EfficientNet_B3 (norm)	100.0	26.00	1.00	1.00	1.00
MobileNetV2	99.95	20.42	1.00	1.00	1.00
MobileNetV2 (norm)	99.95	20.73	1.00	1.00	1.00
ResNeXt50_32X4D	100.0	31.19	1.00	1.00	1.00
ResNeXt50_32X4D (norm)	100.0	31.12	1.00	1.00	1.00

Results – second stage (SphereFace)

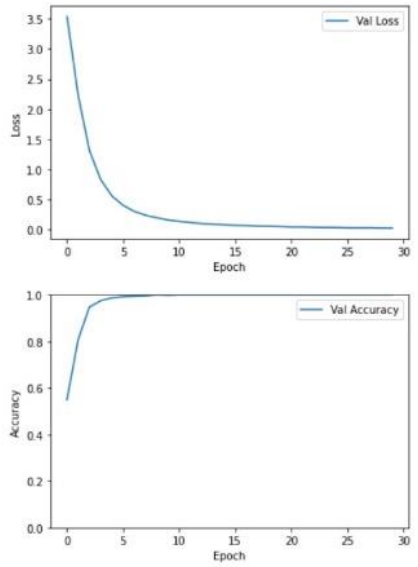
convnext_small



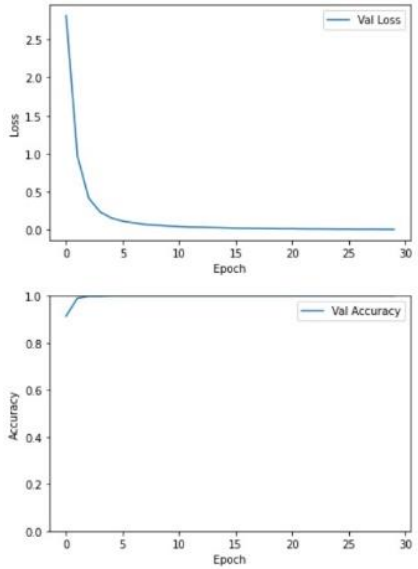
efficientnet_b3



mb_net_v2



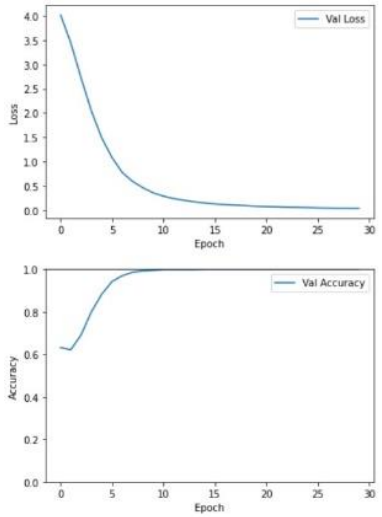
resnext50_32x4d



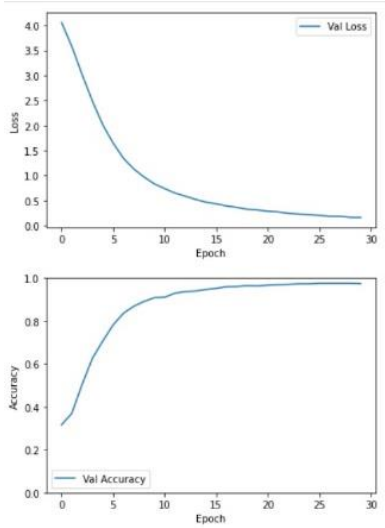
	Accuracy	MEPT	wa precision	wa recal	wa f1 score
ConvNeXt_small	100.0	86.13	1.00	1.00	1.00
ConvNeXt_small (norm)	100.0	86.19	1.00	1.00	1.00
EfficientNet_B3	99.95	26.10	1.00	1.00	1.00
EfficientNet_B3 (norm)	99.95	26.01	1.00	1.00	1.00
MobileNetV2	99.87	19.42	1.00	1.00	1.00
MobileNetV2 (norm)	99.95	19.73	1.00	1.00	1.00
ResNeXt50_32X4D	99.91	31.22	1.00	1.00	1.00
ResNeXt50_32X4D (norm)	99.95	31.19	1.00	1.00	1.00

Results – second stage (ArcFace)

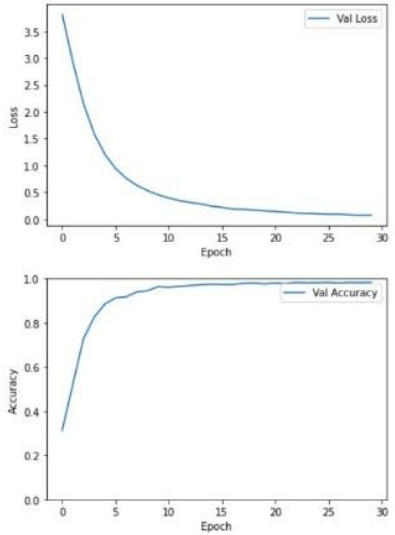
convnext_small



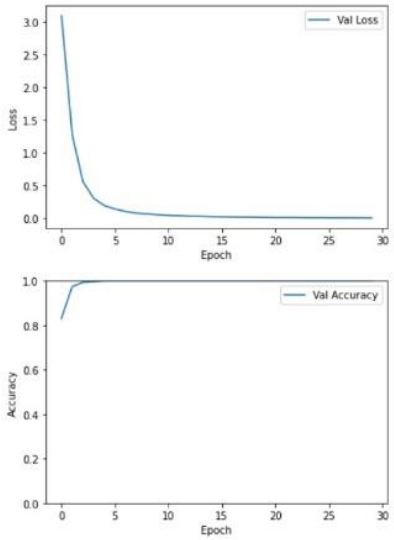
efficientnet_b3



mb_net_v2



resnext50_32x4d



	Accuracy	MEPT	wa precision	wa recal	wa f1 score
ConvNeXt_small	100.0	86.14	1.00	1.00	1.00
ConvNeXt_small (norm)	99.78	86.23	1.00	1.00	1.00
EfficientNet_B3	97.87	26.10	0.98	0.98	0.98
EfficientNet_B3 (norm)	97.62	26.01	0.98	0.98	0.98
MobileNetV2	98.45	19.62	0.99	0.99	0.99
MobileNetV2 (norm)	98.49	19.68	0.99	0.99	0.99
ResNeXt50_32X4D	99.91	31.28	1.00	1.00	1.00
ResNeXt50_32X4D (norm)	100.0	31.26	1.00	1.00	1.00

Stage 2 – summary

On the first stage backbone network with ImageNet weights was used for embedding extraction and on the second stage backbone network trained on full DoctorP disease dataset was used for embedding extraction.

The classification part of the networks was added: two linear layers joined with a ReLU activation function and DropOut(0.2). The CrossEntropy loss function and Adam optimizer with a learning rate of 0.0001 were used for training the networks for 30 epochs with a batch size of 64. Evaluation parameters obtained after 5 runs are presented in Table

NORM	ConvNeXt_small		EfficientNet_B3		MobileNetV2		ResNeXt50_32X4D	
	Accuracy	MEPT	Accuracy	MEPT	Accuracy	MEPT	Accuracy	MEPT
Contrastive	88.03%	148.43	**95.15%**	45,52	90.94%	21,32	88.80%	57,34
Triplet	92,56%	214,31	**95,73%**	67,56	90.89%	52,93	89,56%	82,63
Quadruple	96.75%	279.41	**96,93%**	86,87	93,79%	68,91	93,79%	107,16
SphereFace	**100,00%**	86.17	**100,0%**	26,00	99,95%	20,73	**100.0%**	31,19
CosFace	**100.0%**	86.19	99,95%	26,01	99,95%	19,73	99,95%	31.22
ArcFace	99.78%	86.23	97,62%	26,02	98.49%	19,68	**100,0%**	31.28

	ConvNeXt_small		EfficientNet_B3		MobileNetV2		ResNeXt50_32X4D	
	Accuracy	MEPT	Accuracy	MEPT	Accuracy	MEPT	Accuracy	MEPT
Contrastive	87.89%	150.03	**93.62%**	45,54	92.53%	22,42	87.85%	57,02
Triplet	92,88%	213,62	**95,33%**	67,13	90.21%	50,79	88,67%	83,34
Quadruple	96.12%	279.41	**97,33%**	86,85	92,87%	68,91	92,51%	107,34
SphereFace	99,87%	86.05	99,95%	26,03	99,95%	20,42	**100.0%**	31,19
CosFace	**100.0%**	86.13	99,95%	26,10	99,87%	19,42	99,91%	31.22
ArcFace	**100.0%**	86.14	97,62%	26,05	98.45%	19,62	99,91%	31.28

Stage 2 – Validation of the generalization abilities (150 images)

Superiority of the Cosine-based loss function in training time and accuracy is clearly seen. Still the generalization ability and effect of normalization is not clear. In table 10 presented results of evaluation of models on **dataset consisting of 150 hard-cases images.**

	ConvNeXt_small (191.7 Mb)		EfficientNet_B3 (47.2 Mb)		MobileNetV2 (13.6 Mb)		ResNeXt50_32X4D (95.8 Mb)	
	Vanilla	Norm	Vanilla	Norm	Vanilla	Norm	Vanilla	Norm
Contrastive	61. 5	59.75	**68.5**	65.75	55.5	56.5	65.33	63.5
Triplet	66	**71**	69.5	70.5	56	56.5	66.5	65.33
Quadruple	72.5	**73.33**	72.5	72.33	60.5	59.75	69.75	71.5
SphereFace	77	76.5	75.75	**78.33**	63.5	63.5	73.5	73
CosFace	**78**	76.33	75.33	75	64.33	65	74.33	75
ArcFace	73.5	**75**	**75**	73.33	61.75	62.5	72.5	73.33

Результаты

1. Угловые функции минимизации потерь показывают лучшие показатели по точности и потребляют меньше ресурсов в процессе обучения
2. Размер сети (количество весов) не гарантирует лучшие результаты. EfficientNet_B3 в 3.5 раза меньше чем ConvNeXt_Small, тем не менее демонстрирует сравнимые показатели
3. Нормализация изображений не дает яркого видимого эффекта на показатели моделей. Тем не менее с учетом затрат на реализацию ее использование видится целесообразным.

Plans

1. Влияние аугментации данных на показатели моделей
2. Применение Attention Mechanism для задач классификации изображений
3. Использование semi-supervised learning для кластеризации изображений

The General Disease Model and Crop Model have been updated!

EfficientNet_B3 + SphereFace + Normalization

N:144521 (2024-05-31 06:00:04) PDD

Прогноз модели: **Общая модель**: Нехватка элементов, Же
{"success": "true", "cropse1":"avocado", "cropse2":"mango", "cropse3":"tomatoes

N:144591 (2024-05-31 08:13:20) PDD IOS

Прогноз модели: **Общая модель**: Тля последствия, Апоплексия | **смор**
{"success": "true", "cropse1":"currant", "cropse2":"blackberry", "cropse3":"grape", "full1":"Aphid ef
"cropse3_disease2":"Black rot", "cropse3_disease3":"Healthy"}

N:144348 (2024-05-30 16:29:08) PDD

Прогноз модели: **Общая модель**: Паутинный клещ, Трипс | **огурц**
{"success": "true", "cropse1":"cucumbers", "cropse2":"currant", "cropse3":"begonia", "full1"
"cropse2_disease2":"Powdery mildew", "cropse2_disease3":"Aphid effects"}

Спасибо за внимание!

email: auzhiskiy@jinr.ru
https://t.me/bigzmey