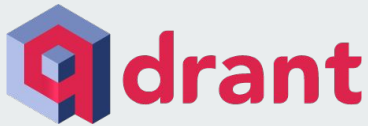
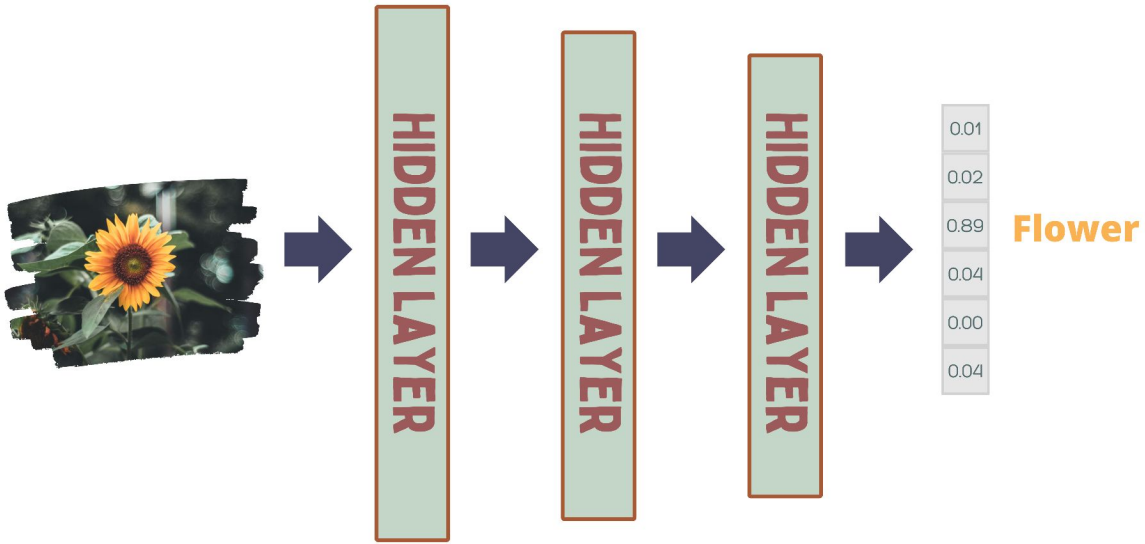
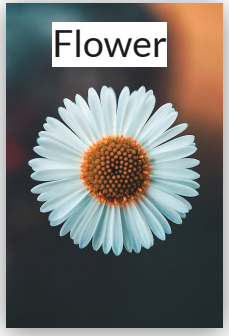
Lowering the entry threshold for Neural Vector Search by applying Similarity Learning

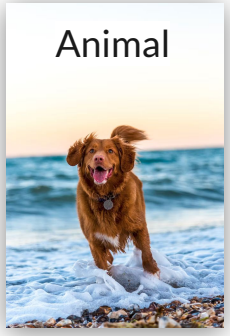
Kacper Łukawski
Developer Advocate
Qdrant

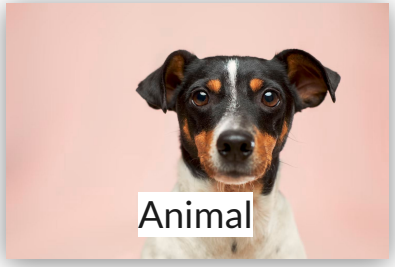


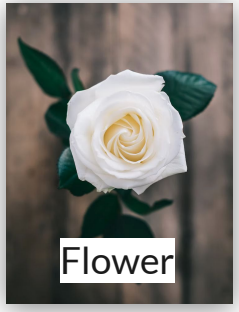
Good old times: Deep Learning for classification

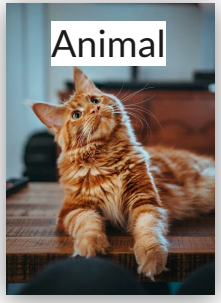








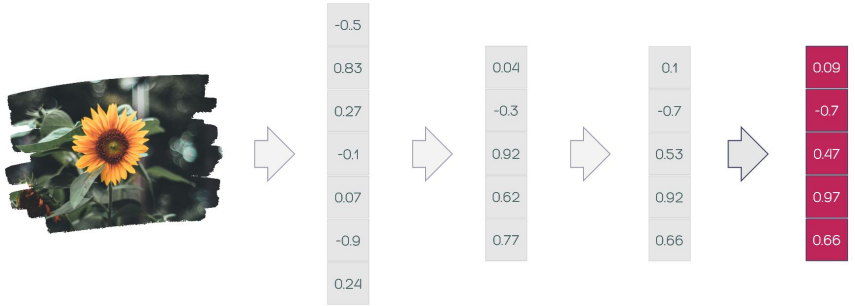




A classification dataset consists of examples and corresponding labels.

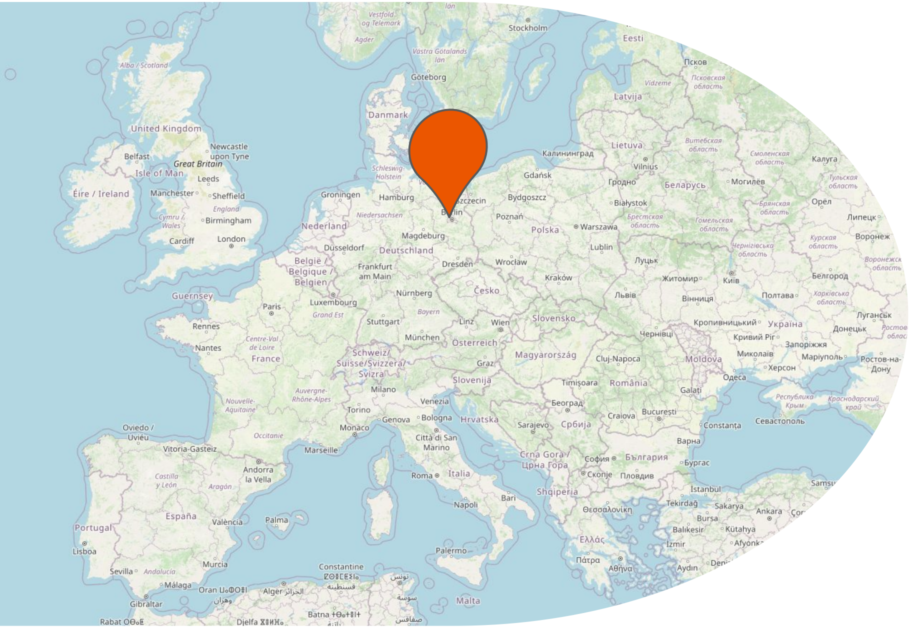
Reminder: how does the neural network work?

At each consecutive network layer, our input data is represented by a vector of the dimensionality determined by the layer size (number of neurons). Each of the vectors might be thought of as some kind of representation of the input, so called **encoding** or **embedding**.



What encodings may remind us about?

Berlin, [52.520008, 13.404954]



Neural embeddings

- They represent input data in N-dimensional latent space
- Two different embeddings should be **close to each other if they represent a similar input object**
- The distance between embeddings might be calculated in various ways (quite commonly with cosine or euclidean distance)

Neural embeddings: office emails

Hope this helps

[0.1, 0.32, 0.4, 0.41, 0.9, 0.5, 0.4]

Just stop bothering me

[0.09, 0.31, 0.41, 0.39, 0.89, 0.51, 0.39]

Neural embeddings: office emails

Thanks in advance

[0.71, 0.62, 0.1, 0.23, 0.8, 0.39, 0.9]

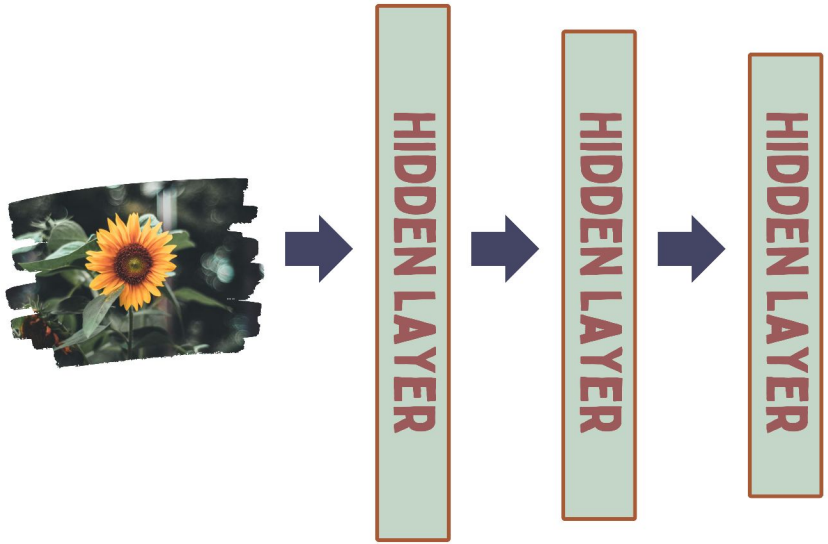
I'm already thanking you for doing me this favor, even though you haven't yet agreed to it. Therefore, you must do it.

[0.71, 0.6, 0.11, 0.24, 0.78, 0.36, 0.91]

Similarity Learning

Similarity Learning: network structure

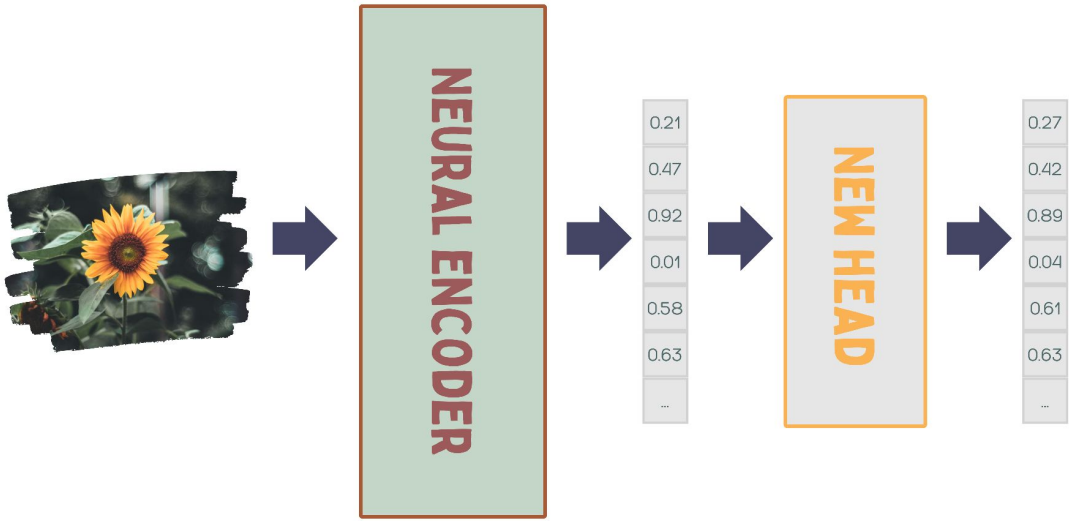
If we have a pretrained network trained for classification, we can simply remove the last softmax-like layer and take the intermediate state as a representation of our data.



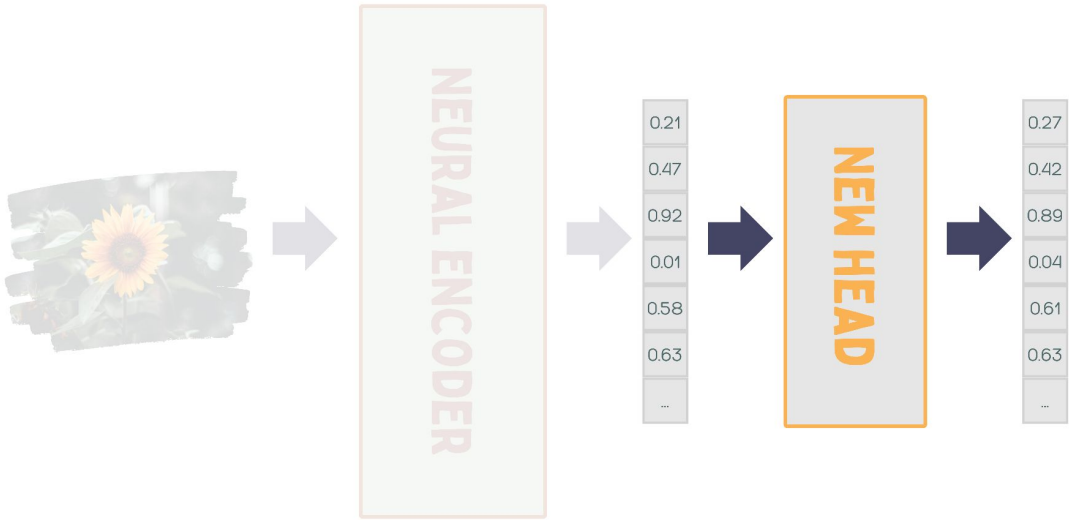
Similarity Learning: pitfalls

- Pretrained models **rarely provide great embeddings** out of the box.
- If the original model was trained i.e. on ecommerce fashion data, then it may struggle with recognizing tumours on medical images.
- The publicly available models, like **BERT**, are trained on datasets like Wikipedia, so may lack some domain specific concepts.
- Training the network from scratch is also time and cost intensive.

Similarity Learning: fine-tuning



Similarity Learning: fine-tuning

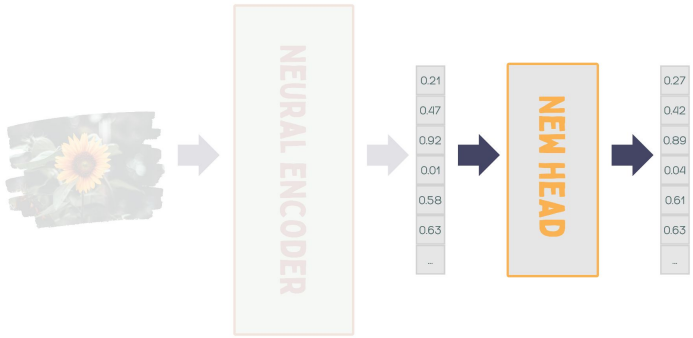


Freezing the parameters of the neural encoder is a common strategy to avoid fine-tuning the whole network. That might cost a little fortune and takes quite a lot of time.

Pretrained models

In practice, the neural encoders have millions of parameters and we don't want to change them that much or even don't want to change them at all. That's why we put an additional layer on top of it and train the new head layers only to solve a particular problem. If the original encoder was trained on images to recognize a particular set of classes, then there is a high chance it will be also able to recognize some others after a little **fine-tuning**.

Pretrained models



Freezing the network

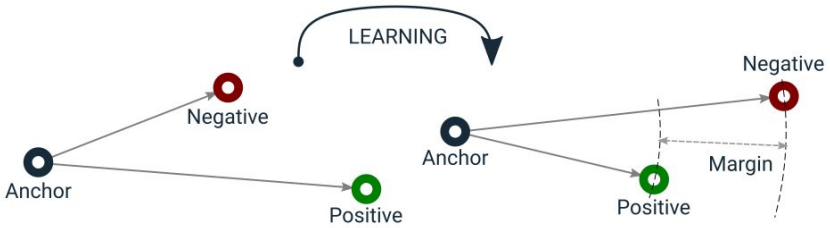
If we freeze the neural encoder, the original part of the network becomes **deterministic**. But in the tutorials you would be still encouraged to load the images again and again in every single epoch. That means a huge overhead and slow training, even on a decent GPU.

Similarity Learning: pitfalls

- In case of classification/regression, we know the target output
- With similarity learning we cannot provide the targets, as the perfect embeddings are **unknown**.

Similarity Learning: the process

We can **fine-tune** our existing network to work better in the new domain. The goal is to **adjust** the embeddings, not change them completely!



Similarity Learning: loss functions

Triplet loss is the most commonly known loss function used in similarity learning. It requires four elements:

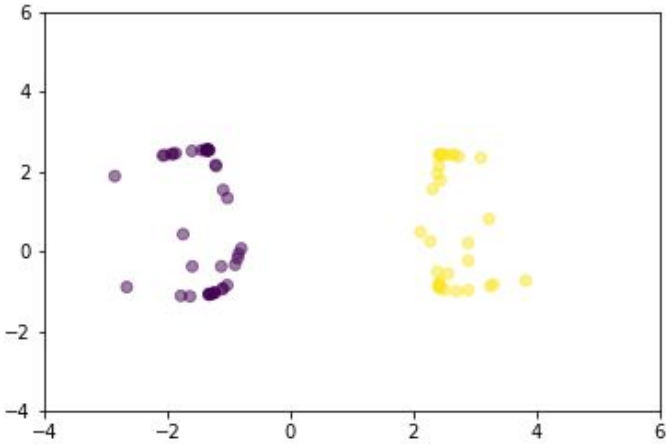
1. Anchor
2. Positive example
3. Negative example
4. Margin

$$L = max(d(a, p) - d(a, n) + m, 0)$$

Source: https://towardsdatascience.com/triplet-loss-advanced-intro-49a07b7d8905

Similarity Learning: pitfalls

Triplet Loss suffers from what is called **vector collapsing**, a common problem in similarity learning. A collapse of the vector space is the state when an encoder satisfies the loss function by simply **mapping all input samples onto a single point** (or a very small area) in the vector space without truly learning useful features for the task.

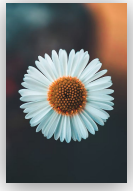
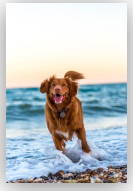


Source: https://towardsdatascience.com/triplet-loss-advanced-intro-49a07b7d8905

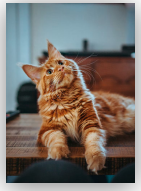
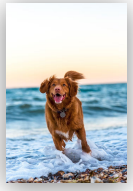
Similarity Learning: dataset structure (groups)

Similarity Learning: dataset structure (pairs)

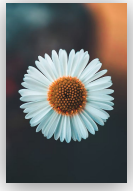

Similarity: **0.7**

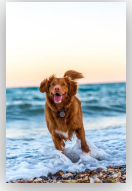
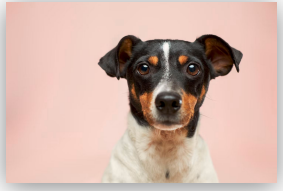
Similarity: **0.0**

Similarity: **0.39**




Similarity: **0.53**

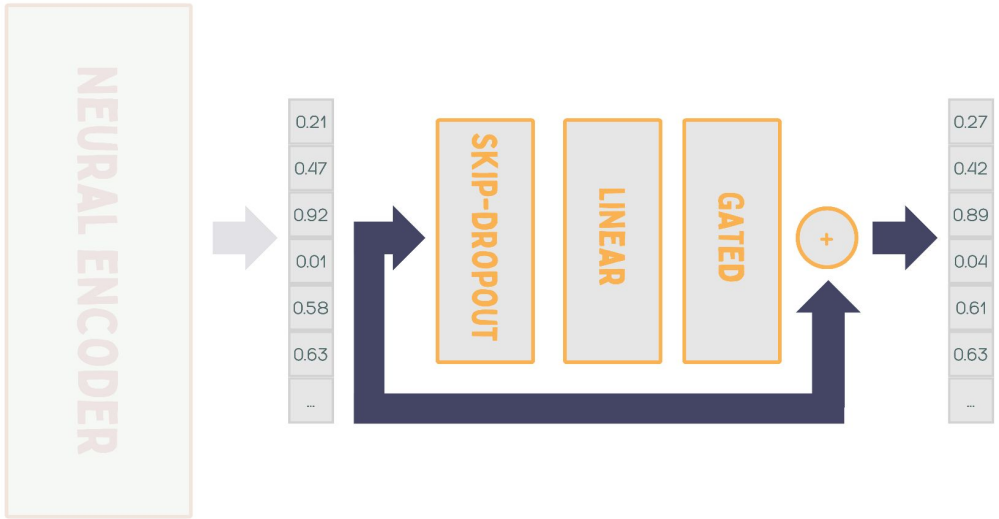



Similarity: **0.82**

Similarity Learning: pitfalls

- Even though the representation might not be that accurate, **it still can capture some data regularities**. Thus, we just want to slightly adjust the embeddings for the new domain.
- In other words, we need to avoid **catastrophic forgetting**.

Similarity Learning: skip connection head



If we just randomly initialize the head layer, it destroys the useful signal that is coming from the main encoder.

This may lead to several problems such as overfitting on the training dataset, being stuck in a local minimum, and unstable loss values among others, failing to effectively tune the parameters of the base model.

Gated layer **has zeros at the very beginning**, so literally starts from the original embeddings.

Quaterion

A PyTorch-Lightning compatible similarity learning framework

1. Built-in caching mechanism for neural embeddings (even up to 100x faster training).
2. The most popular similarity learning loss functions already built-in.
3. Skip-connection head layer available.
4. Vector collapse prevention in triplet loss.
5. And many more…

https://quaterion.qdrant.tech/

Moving to production

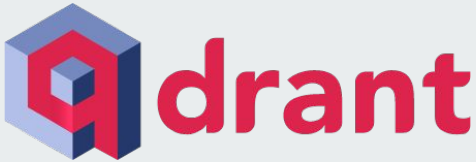
Similarity Learning: use cases

1. Anomaly detection
2. Recommendation systems
3. Classification, even the extreme case
4. Question answering
5. **Semantic search**
6. And much more…

Problems with using Similarity Learning at scale

- Since our network does no longer produce the probability distributions over classes, but some high-dimensional points in space, we cannot derive the predictions directly from them.
- Thus, we need to find the closest points in that space and need to do it efficiently.
- A naive implementation of kNN solves this problem, but has linear complexity.
- There has been a lot going on in the area of **Approximate Nearest Neighbours**.

Vector Search in production



Qdrant is a vector search database using **HNSW**, one of the most promising algorithms for Approximate Nearest Neighbours.

- It's written in Rust and offers great performance.
- Allows to interact by HTTP or gRPC protocols (also by official Python/Rust/Go clients).
- Runs both in single and multiple node setup.
- Incorporates category, geocoordinates and full-text filters to the vector search.

Questions?

Kacper Łukawski
Developer Advocate
Qdrant

https://www.linkedin.com/in/kacperlukawski/
https://twitter.com/LukawskiKacper
https://github.com/kacperlukawski