Exploring Deep Learning Methods for Medical Image Tagging

Foivos Charalampakos
M.Sc. Thesis – M.Sc. Computer Science
Department of Informatics, AUEB

Presentation outline

1. Introduction
2. Data
3. Methods & Results
4. Takeaways & Future work

1. Introduction
2. Data
3. Methods & Results
4. Takeaways & Future work

Image Classification

- Image classification is the task of assigning a label or category (class) to an entire image

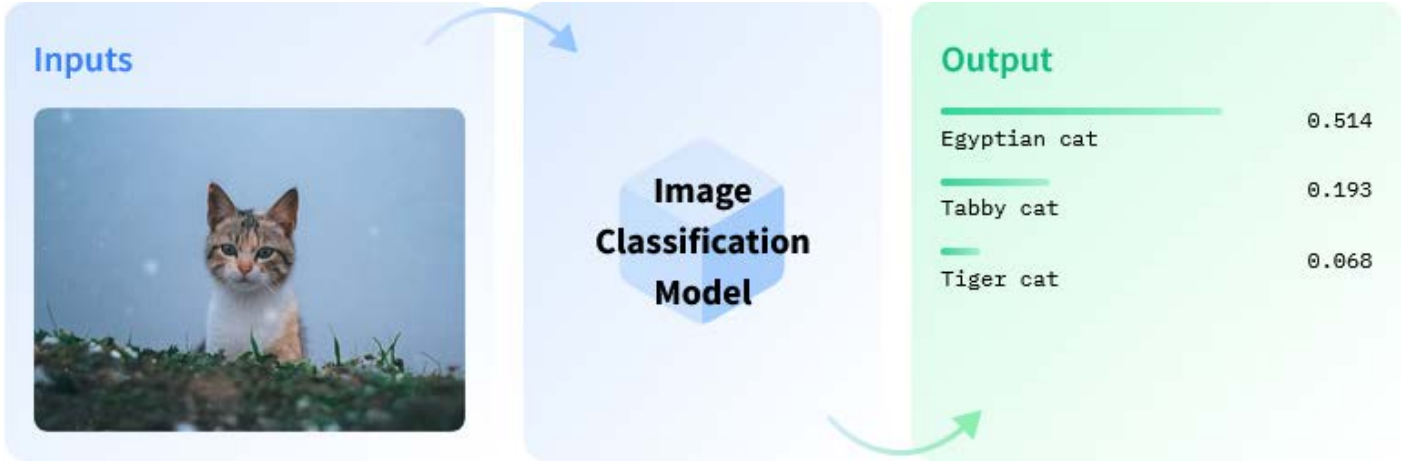


Figure taken from https://huggingface.co/tasks/image-classification

Multi-Label Image Classification

- In multilabel classification, an image can be associated with more than one label and model predicts the probability of each label independently

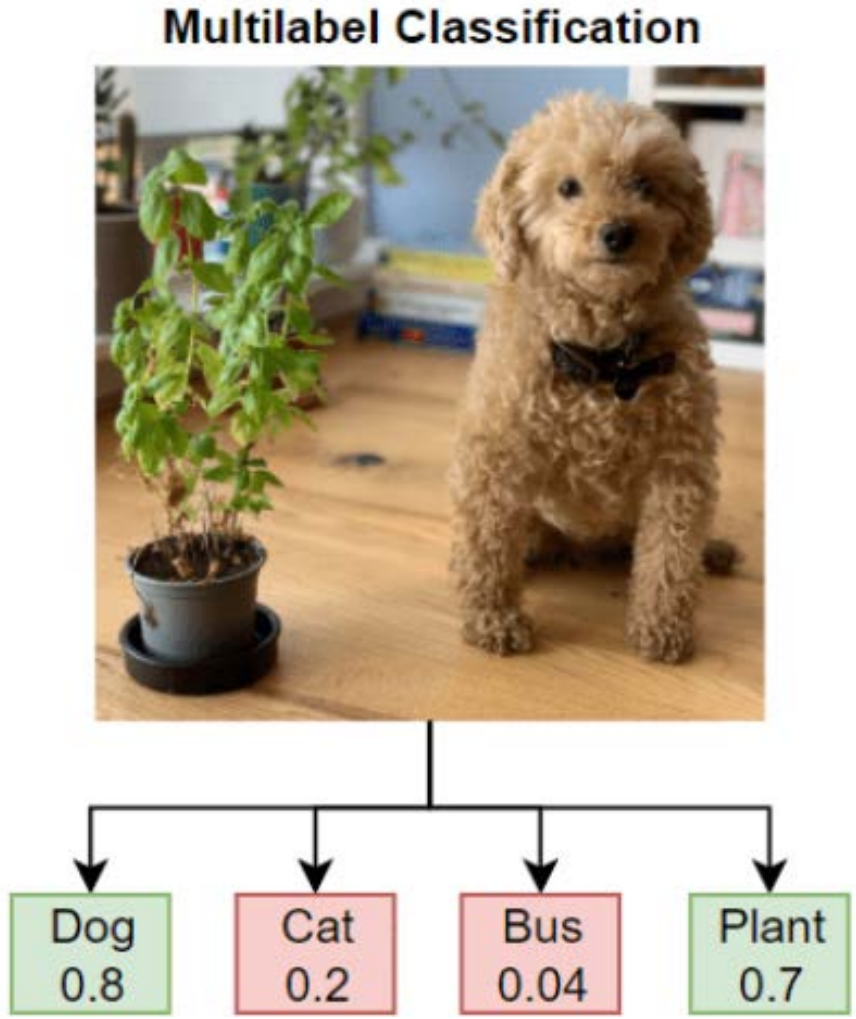
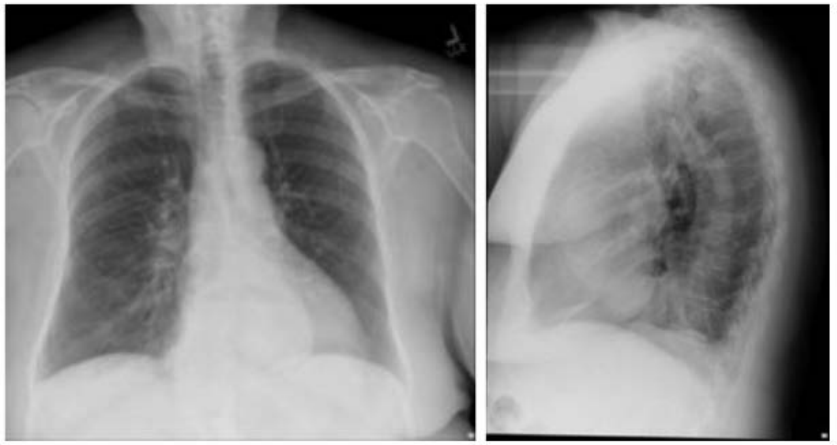


Figure taken from https://www.mathworks.com/help/deeplearning/ug/multilabel-image-classification-using-deep-learning.html

Medical Image Tagging

- Multi-Label classification task
- Identify and assign one or more medical terms (tags) that correctly describe body organs, possible findings etc. on a given image



FINDINGS: No change. No visible active cardiopulmonary disease. Both lungs remain clear and expanded. Heart and pulmonary XXXX are normal. No change in the large hiatus hernia.

SYSTEM TAGS: hiatus hernia / Hernia, Hiatus

HUMAN TAGS: Hernia, Hiatal/ large

Figure taken from J. Pavlopoulos et al. "Diagnostic captioning: a survey". In: Knowl Inf Syst 64, pp. 1691–1722 (2022)

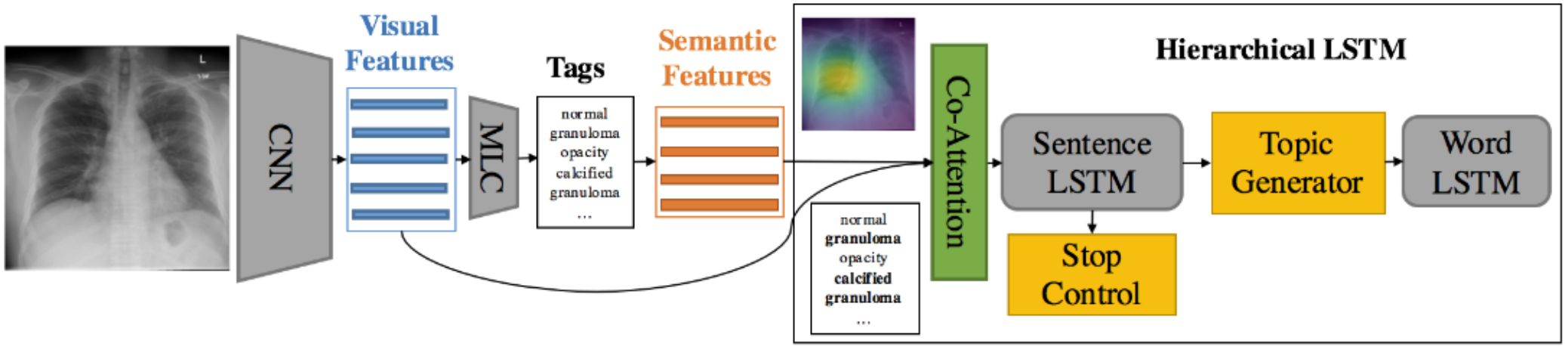
Why is it important?

- Clinicians examine and analyze a large number of medical images (i.e., MRI scans, X-rays and PET/CT scans) in their daily workflow
  - Time consuming
  - Present difficulties for inexperienced clinicians

Why is it important? (cont.)

- Medical Image Tagging can potentially accelerate the diagnostic process by providing possible findings present in an image and help clinicians by reducing their workload
- It can constitute a first step in the automatic development of draft medical reports where the predicted tags could be used as keywords in a coherent textual diagnosis (diagnostic captioning)

Captioning system example



1: B. Jing et al. "On the Automatic Generation of Medical Imaging Reports". In: *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)* (2018). pp. 2577–2586.

~~1. Introduction~~

2. Data

3. Methods & Results

4. Takeaways & Future work

2022 ImageCLEFmedical Caption

- ImageCLEF[1] is an annual campaign of competition tasks
  - ImageCLEFmedical revolves around the processing and understanding of medical images
  - The 2022 campaign included an Image Captioning task and a Tuberculosis analysis task
  - The Captioning task consisted of two sub-tasks:
    - Concept Detection
    - Caption Prediction
  - The thesis focused on the Concept Detection sub-task

1: https://www.imageclef.org/

Dataset

- A set of 90,920 radiology images (e.g., X-rays, CT-Scans)
- Subset of the ROCO dataset[1]
  - Images extracted from PubMed Central[2]
- 8,374 unique tags (here called concepts)
- The data was split into training (65%), validation (15%) and development (20%) sets
  - There was an official test set of 7,601 images but its annotations were not provided so the development set was used as a private test set in the experiments

1: O. Pelka et al. "Radiology Objects in COntext (ROCO): A Multimodal Image Dataset". In: vol. 11043. Lecture Notes in Computer Science (LNCS), 2018, pp. 180–189.

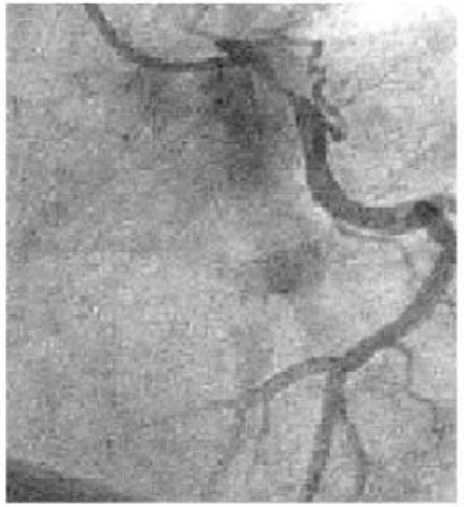
2: https://www.ncbi.nlm.nih.gov/pmc/

Dataset (cont.)

CUI: C1306645
UMLS Term: Plain X-ray

CUI: C0037303
UMLS Term: Bone structure of cranium

CUI: C0029053
UMLS Term: Decreased translucency



CUI: C0002978
UMLS Term: angiogram

CUI: C0678234
UMLS Term: Stenosis Morphology

CUI: C0162577
UMLS Term: Angioplasty

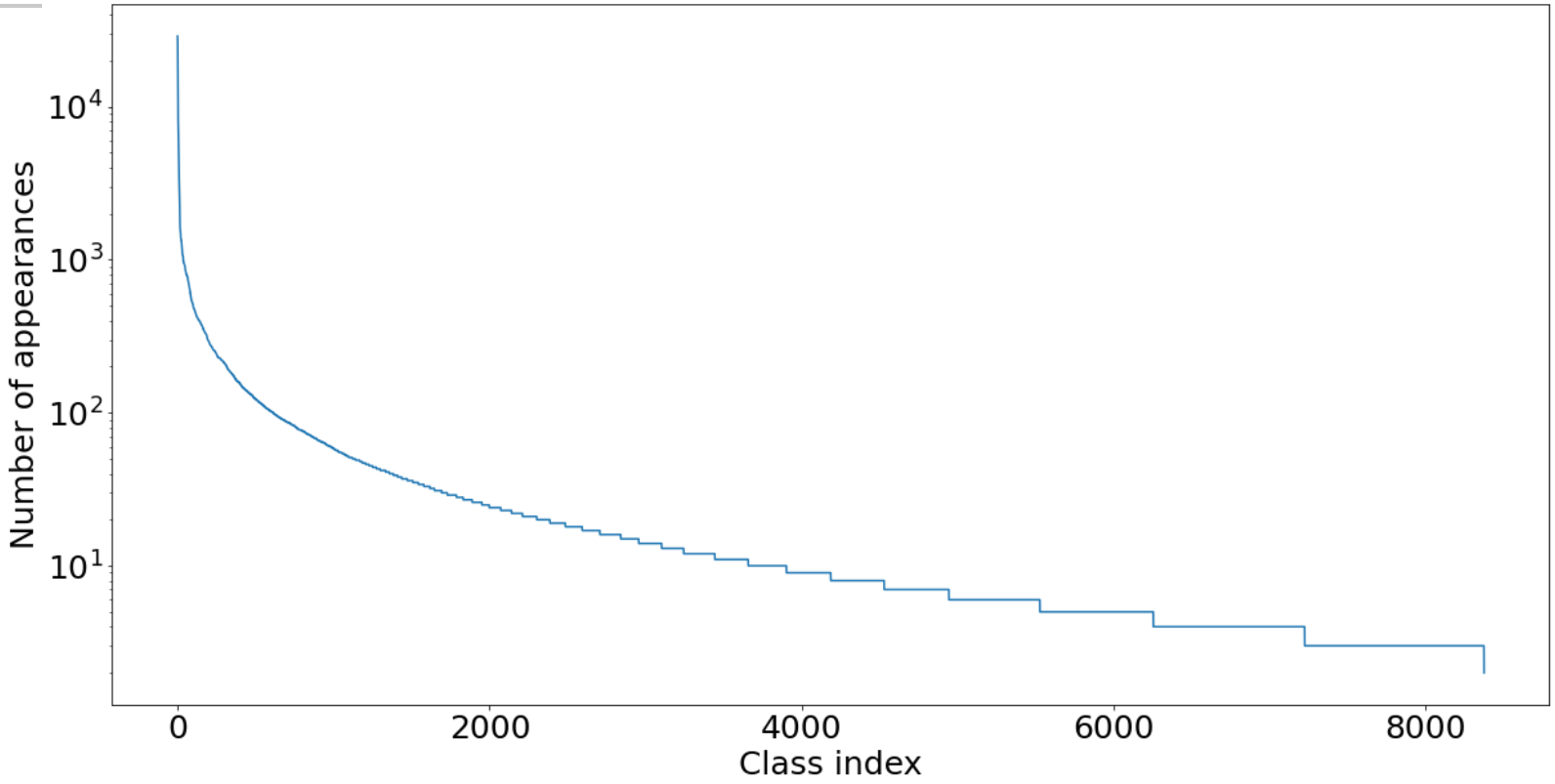
CUI: C0040405
UMLS Term: angiogram

Most common concepts

CUI	UMLS term	Images
C0040405	X-RAY COMPUTED TOMOGRAPHY	28,885
C1306645	PLAIN X-RAY	26,412
C0024485	MAGNETIC RESONANCE IMAGING	15,693
C0041618	ULTRASONOGRAPHY	12,236
C0817096	CHEST	8,030
C0002978	ANGIOGRAM	6,464
C0000726	ABDOMEN	6,243
C0037303	BONE STRUCTURE OF CRANIUM	5,175
C0221198	LESION	4,094
C0205131	AXIAL	3,528

Concepts occurrence distribution

- Long-tail
- Almost half of the concepts (4,716) appear less than 10 times
- 1,149 concepts appear only 3 times



Evaluation measure

- F_1 score averaged over the evaluation instances
  - 'samples' average
- Individual scores are computed using the true and predicted multi-hot vectors of each instance
  - True Positives (TP), False Positives (FP) and False Negatives (FN) are calculated per instance, not class

$$F_1^* = \frac{\sum_{i=1}^{n} F_1(y^i, \hat{y}^i)}{n}, \qquad n = number\ of\ instances$$

~~1. Introduction~~

~~2. Data~~

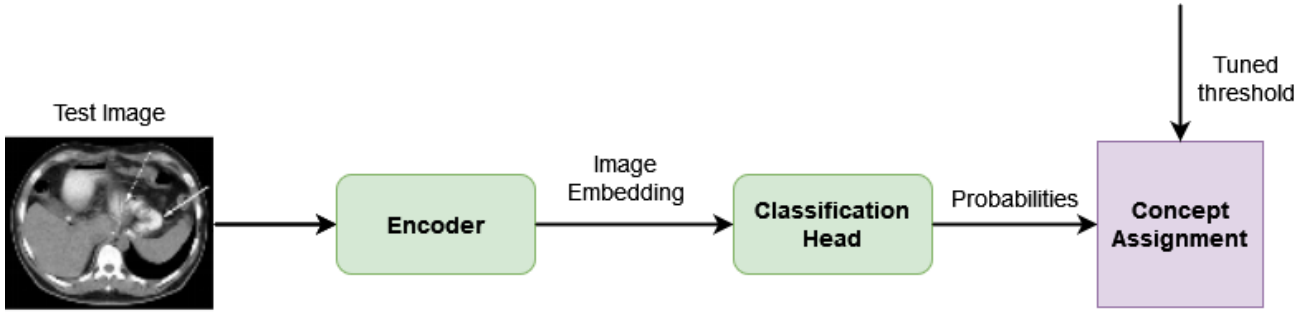
3. Methods & Results

4. Takeaways & Future work

Neural Classification methods

Baseline

- Image encoder + Feed-Forward classification head



- The image encoder outputs an image embedding (a feature vector) $v \in \mathbb{R}^D$
  - This vector is produced by squashing the spatial dimensions via a pooling operation (e.g., average, max etc.)
- At inference, a label was assigned to the test image if its output probability exceeded a threshold t.
  - $p_{label} \geq t$
- Instead of using the default value $t = 0.5$, we tuned t using the validation data

Baseline: visual encoder

- Experiments with various encoders
  - Convolutional Neural Networks (CNN)
  - Vision Transformer (ViT)
  - Hybrids

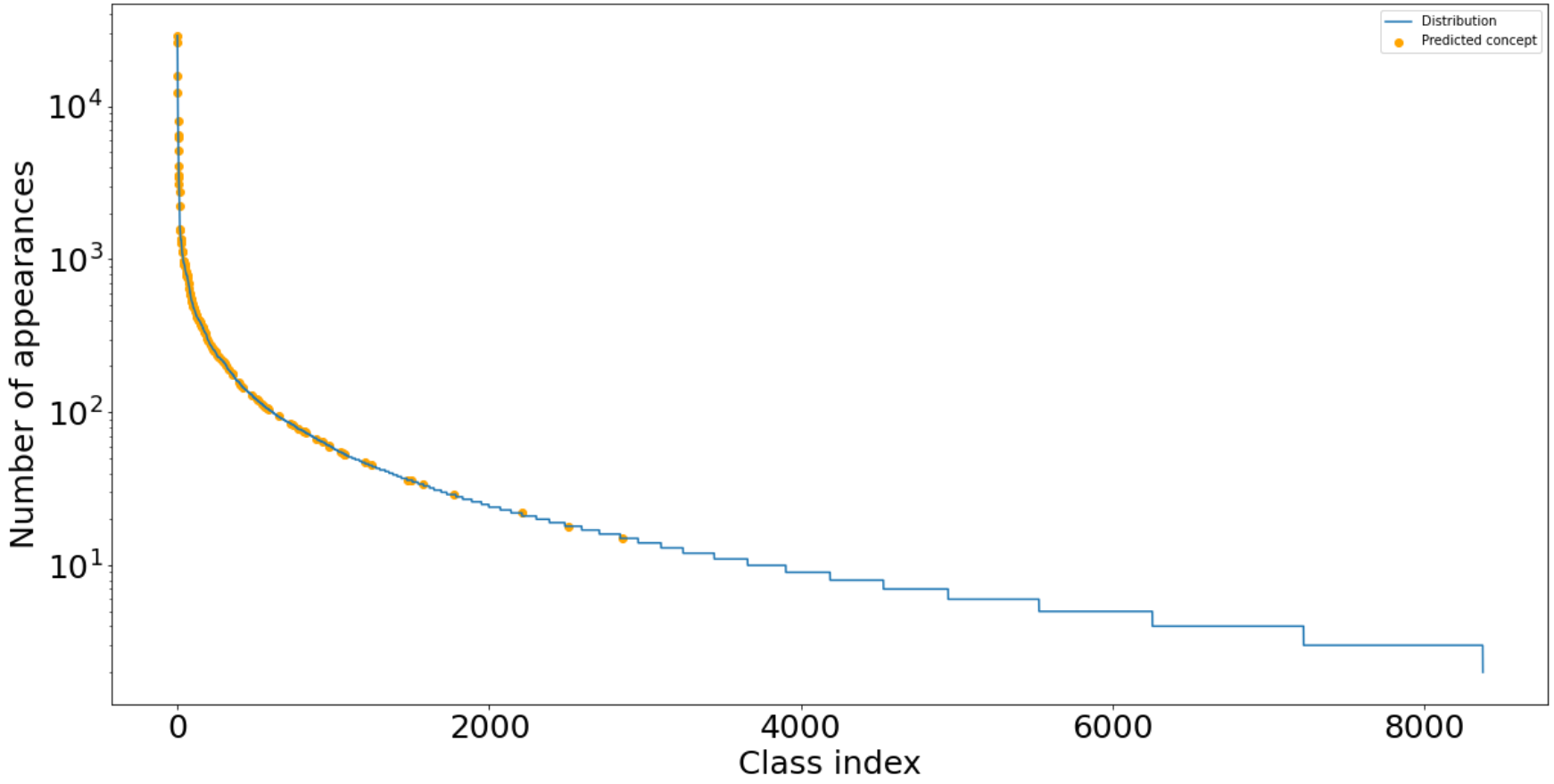
Baseline: visual encoder – Results

- Larger and more complex architectures are not necessarily better
- CNNs outperformed ViT-based models
  - ViTs are data-hungry
  - They lack the intrinsic biases in modeling local visual structures
    - They implicitly learn such biases from large-scale data
- Using a linear classification head produced better scores

Encoder	# Parameters	F_1^* Score
EfficientNetV2-B0[TL21]	7.2M	**45.25**
DenseNet-121[Hua+17]	8.1M	44.08
ViT-B[Dos+21]	86M	39.86
CoAtNet-0[Dai+21]	25M	43.26
NFNet-F1[Bro+21]	132.6M	45.18

Approach	F_1^* Score
EfficientNetV2-B0[TL21] + MLP	45.05
EfficientNetV2-B0[TL21] + LR	**45.25**

Predictions analysis



Baseline: loss function

- Loss used so far: Binary Cross-Entropy

$$\mathcal{L}_{BCE_c} = -\left(y_c \log(p_c) + (1 - y_c) \log(1 - p_c)\right)$$

- Most positions of the gold multi-hot vector are usually filled with zeros
- The loss tends to zero-out most labels to have a small value

Baseline: loss function – Focal loss

- Focal loss:[1]

$$\mathcal{L}_{FL_c} = -\left(y_c \alpha (1-p_c)^{\gamma} \log(p_c) + (1-y_c)(1-\alpha) p_c^{\gamma} \log(1-p_c)\right)$$

- α, γ are tunable hyper-parameters that control the behavior of the loss value
  - $\alpha \in [0, 1]$ is a weighting factor which balances the importance of positive and negative examples
  - $\gamma \geq 0$ differentiates between easy and hard examples by down-weighting easy examples and thus focusing the training process on hard-to-classify examples
    - The higher its value, the lower the overall loss for well-classified examples
    - i.e., for $\gamma = 2$, a sample with $y_c = 1$ and $p_c = 0.9$ would have 100x lower loss compared to BCE

1: T. Lin et al. "Focal Loss for Dense Object Detection". In: IEEE International Conference on Computer Vision (ICCV) (2017), pp. 2999–3007

Baseline: loss function – Asymmetric loss

- Asymmetric loss:[1]

$$\mathcal{L}_{ASL_c} = -\left(y_c(1-p_c)^{\gamma+}\log(p_c) + (1-y_c)p_{c_s}^{\gamma-}\log(1-p_{c_s})\right)$$

- Setting high γ in Focal loss to sufficiently down-weight the contribution of easy negatives, may eliminate the contribution from the rare positive samples
  - Decouple the focusing levels of the positive and negative samples.
    - γ_+, γ_- : positive and negative focusing hyper-parameters
- Additional asymmetric mechanism using probability shifting: $p_{c_s} = \max(\mathrm{p} - \mathrm{m}, 0)$
  - m is a tunable hyper-parameter
  - Discards negative samples when their probability is very low
    - Moves the loss function to the right, by a factor $m \rightarrow L_- = 0$ when $p < m$

1: E. Ben Baruch et al. "Asymmetric Loss For Multi-Label Classification". In: IEEE/CVF International Conference on Computer Vision (ICCV) (2021), pp. 82–91.

Baseline: loss function – soft F_1 loss

- Soft F_1 loss:[1]

$$\mathcal{L}_{F_1} = 1 - sF_1 = 1 - 2 \cdot \frac{s\text{Precision} \cdot s\text{Recall}}{s\text{Precision} + s\text{Recall}}$$

- Directly optimizes the evaluation metric
- Differentiable by computing TPs, FPs and FNs using the output probabilities of the model, without applying any threshold to round them to binary decisions

$$s\text{Precision} = \frac{sTP}{sTP + sFP} \qquad s\text{Recall} = \frac{sTP}{sTP + sFN}$$

1: https://towardsdatascience.com/the-unknown-benefits-of-using-a-soft-f1-loss-in-classification-systems-753902c0105d

Baseline: loss function – soft F_1 loss (cont.)

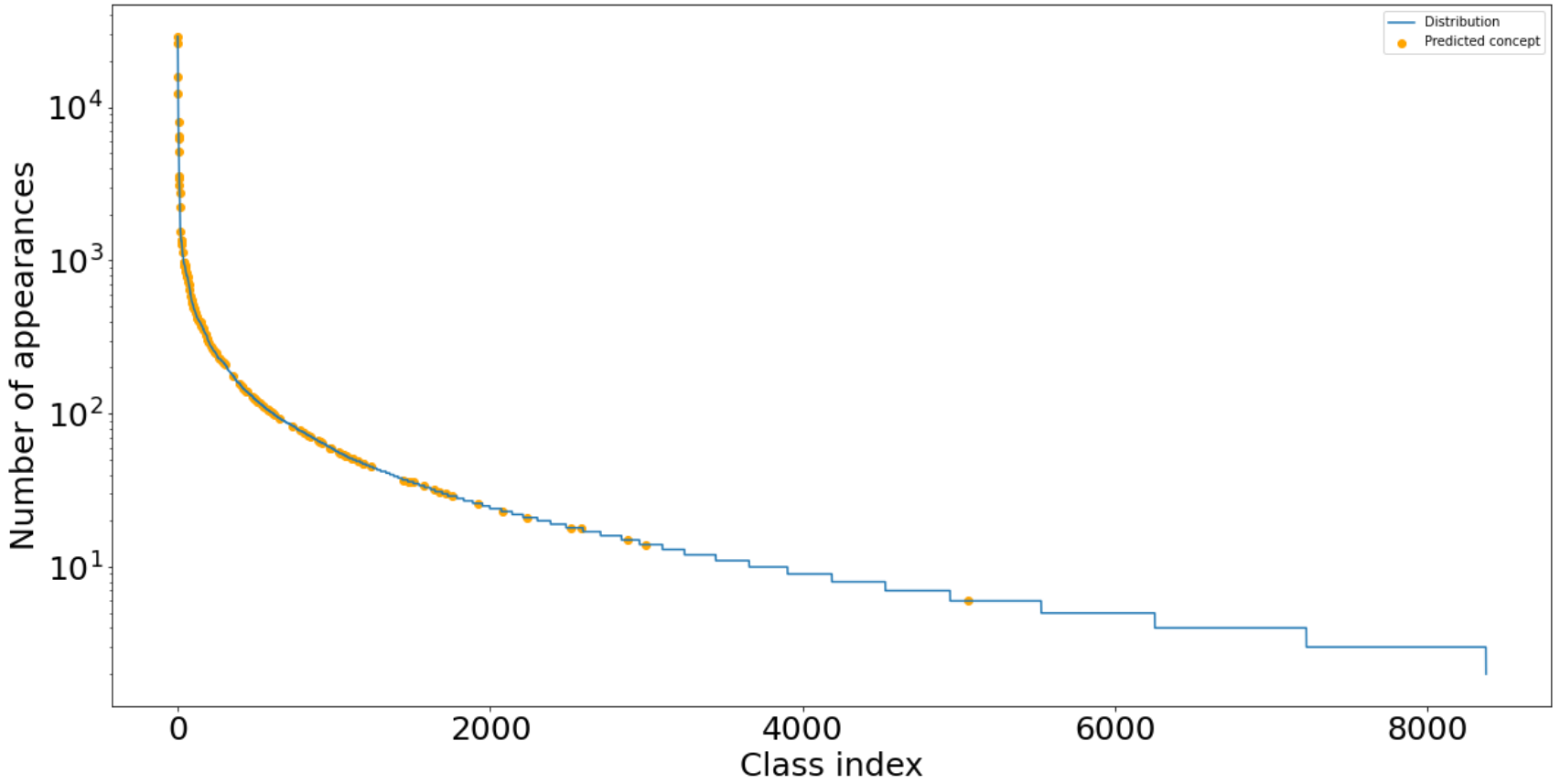
- An example: Movies genre classification
  - $y_{Action} = 1,\ p_{Action} = 0.8$
    - 0.8 x 1 = 0.8 TP (because the target is 1 and the model predicted 1 with 0.8 chance)
    - 0.2 x 1 = 0.2 FN (because the target is 1 and the model predicted 0 with 0.2 chance)
    - 0.8 x 0 = 0 FP (because the target is 1, not valid)
    - 0.2 x 0 = 0 TN (because the target is 1, not valid)
  - $y_{Action} = 0,\ p_{Action} = 0.8$
    - 0.8 x 0 = 0 TP (because the target is 0, not valid)
    - 0.2 x 0 = 0 FN (because the target is 0, not valid)
    - 0.8 x 1 = 0.8 FP (because the target is 0 and the model predicted 1 with 0.8 chance)
    - 0.2 x 1 = 0.2 TN (because the target is 0 and the model predicted 0 with 0.2 chance)

Baseline: loss function – Results

- Current best score: $F_1^* = 45.25$
- $\mathcal{L}_{ASL}$ provided a very slight increase
  - Output probabilities given to each label were higher. However, average performance depends on t
  - More extensive tuning could possibly provide more suitable hyper-parameter values for this specific dataset
- $\mathcal{L}_{BCE} * \mathcal{L}_{F_1}$ provided further increase
  - Experiments with $\mathcal{L}_{F_1}$ did not converge

Loss function	F_1^* Score
$\mathcal{L}_{FL}\ (\gamma = 2, \alpha = 0.25)$	45.14
$\mathcal{L}_{FL}\ (\gamma = 2, \alpha = 0.4)$	44.87
$\mathcal{L}_{FL}\ (\gamma = 4, \alpha = 0.25)$	44.46
$\mathcal{L}_{FL}\ (\gamma = 4, \alpha = 0)$	44.15
$\mathcal{L}_{FL}\ (\gamma = 2, \alpha = 0)$	44.57
$\mathcal{L}_{ASL}\ (\gamma_- = 4, \gamma_+ = 0, m = 0.05)$	45.09
$\mathcal{L}_{ASL}\ (\gamma_- = 2, \gamma_+ = 1, m = 0)$	45.00
$\mathcal{L}_{ASL}\ (\gamma_- = 4, \gamma_+ = 1, m = 0.5)$	43.01
$\mathcal{L}_{ASL}\ (\gamma_- = 10, \gamma_+ = 0, m = 0.05)$	43.98
$\mathcal{L}_{ASL}\ (\gamma_- = 2, \gamma_+ = 0, m = 0.05)$	45.45
$\mathcal{L}_{F_1} * \mathcal{L}_{BCE}$	**45.85**

Baseline: loss function – ASL predictions



Baseline: optimization

- Optimizer so far: SGD and variants[1]

- Generalization can potentially be improved by simultaneously minimizing loss value and loss sharpness

1: D. P. Kingma and J. Ba. "Adam: A Method for Stochastic Optimization". In: 3rd International Conference on Learning Representations (ICLR). San Diego, CA, USA, 2015.

Baseline: optimization – SAM

- Sharpness-Aware Minimization (SAM) seeks parameters that lie in neighborhoods having uniformly low loss value (rather than parameters that only have low loss value themselves)
  - It seeks flatter minima (taking sharpness into consideration)

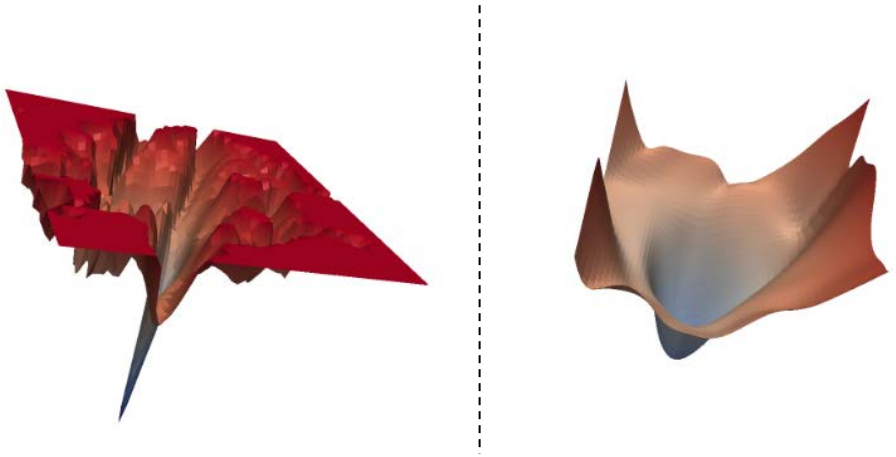


Figure taken from P. Foret et al. "Sharpness-aware Minimization for Efficiently Improving Generalization". In: International Conference on Learning. 2021

Baseline: optimization – SAM (cont.)

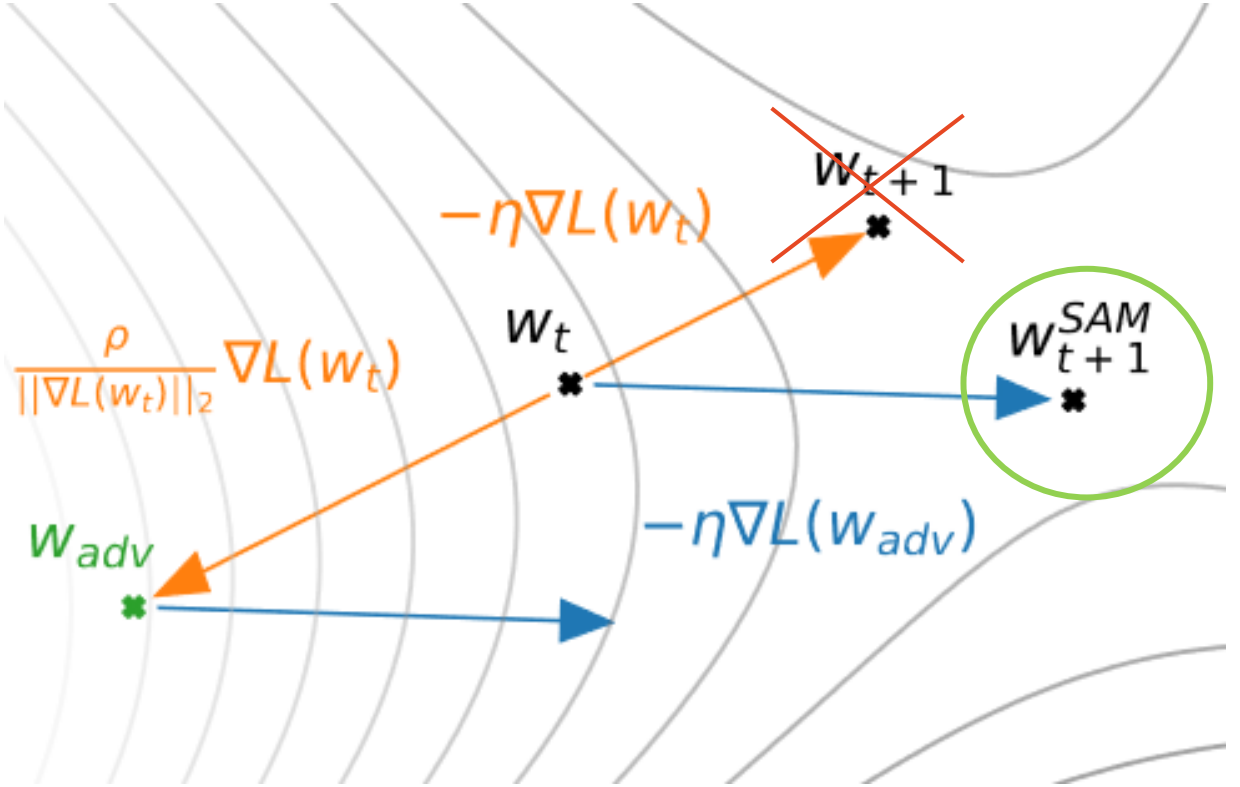


Figure taken from P. Foret et al. "Sharpness-aware Minimization for Efficiently Improving Generalization". In: International Conference on Learning. 2021

Baseline: optimization – Bayesian Optimization

- Tried to tune a separate threshold t_c for every label
- Large size of the label set + large search space
  - Grid Search is infeasible
- We employed Bayesian Optimization (BO)[1] using the evaluation metric as the objective function, aiming at maximizing it

1: J. Snoek et al. "Practical Bayesian Optimization of Machine Learning Algorithms". In: Advances in Neural Information Processing Systems. Vol. 25. Lake Tahoe, NV, USA, 2012.

Baseline: optimization – Results

- Current best score: $F_1^* = 45.85$ $(\mathcal{L}_{F_1} * \mathcal{L}_{BCE})$
- SAM improves generalization by a small margin
- BO did not manage to outperform the performance of the single threshold
  - Few iterations due to time restrictions

Approach	F_1^* Score
$\mathcal{L}_{F_1}$ + SAM	**45.94**
$\mathcal{L}_{F_1}$ + BAYES OPT.	45.68

Adding information from the labels

- Mainly experimented with Graph Neural Networks (GNNs)
  - Created a graph $\mathcal{G}$ based on the statistical co-occurrence of the labels
  - $\mathcal{G}$ was processed using GNNs and the acquired (node/label) features were combined with image features

Adding information from the labels – ML-GCN (objective)

- The final node embeddings $\boldsymbol{H} \in \mathbb{R}^{|C| \times D}$ are combined with the image embedding $\boldsymbol{v}$ to obtain the predicted scores:

$$\hat{y} = \sigma(\boldsymbol{H}\boldsymbol{v})$$

- The two networks can be trained with a supervised loss
- We also performed an ablation test by removing the GCN and directly combining $\boldsymbol{v}$ with the 'initial' node embeddings

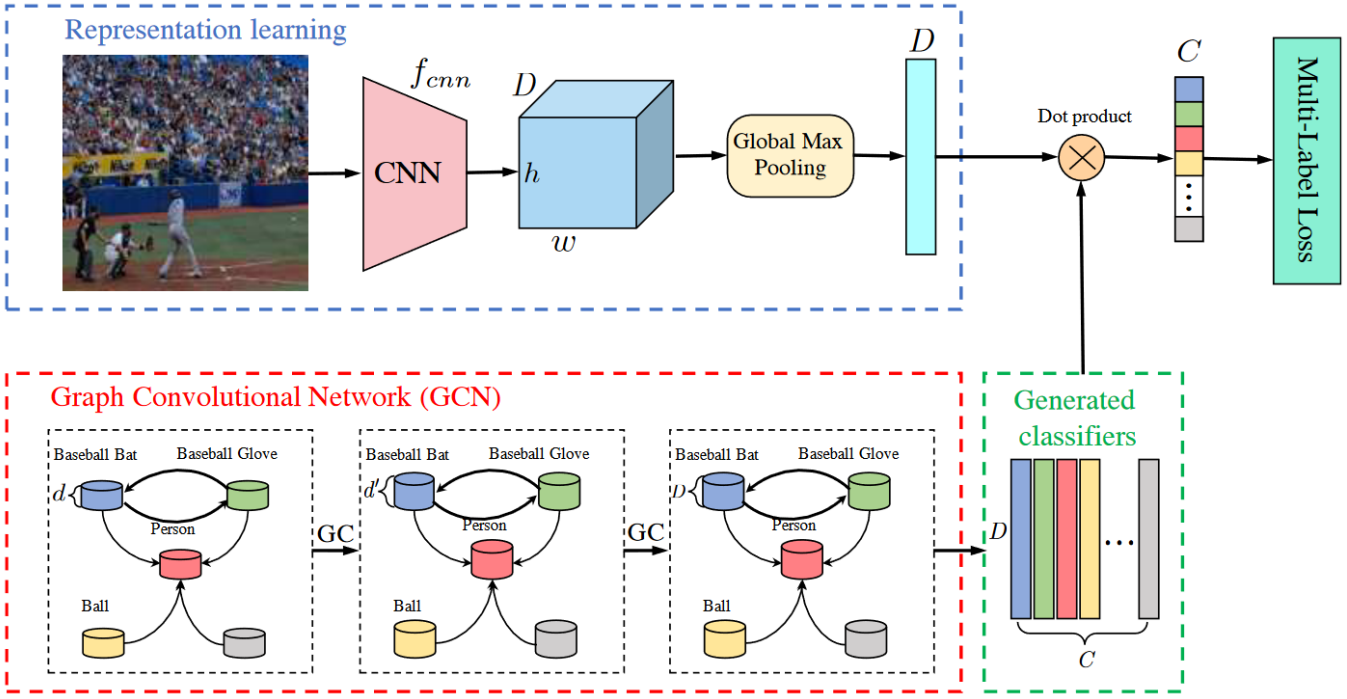


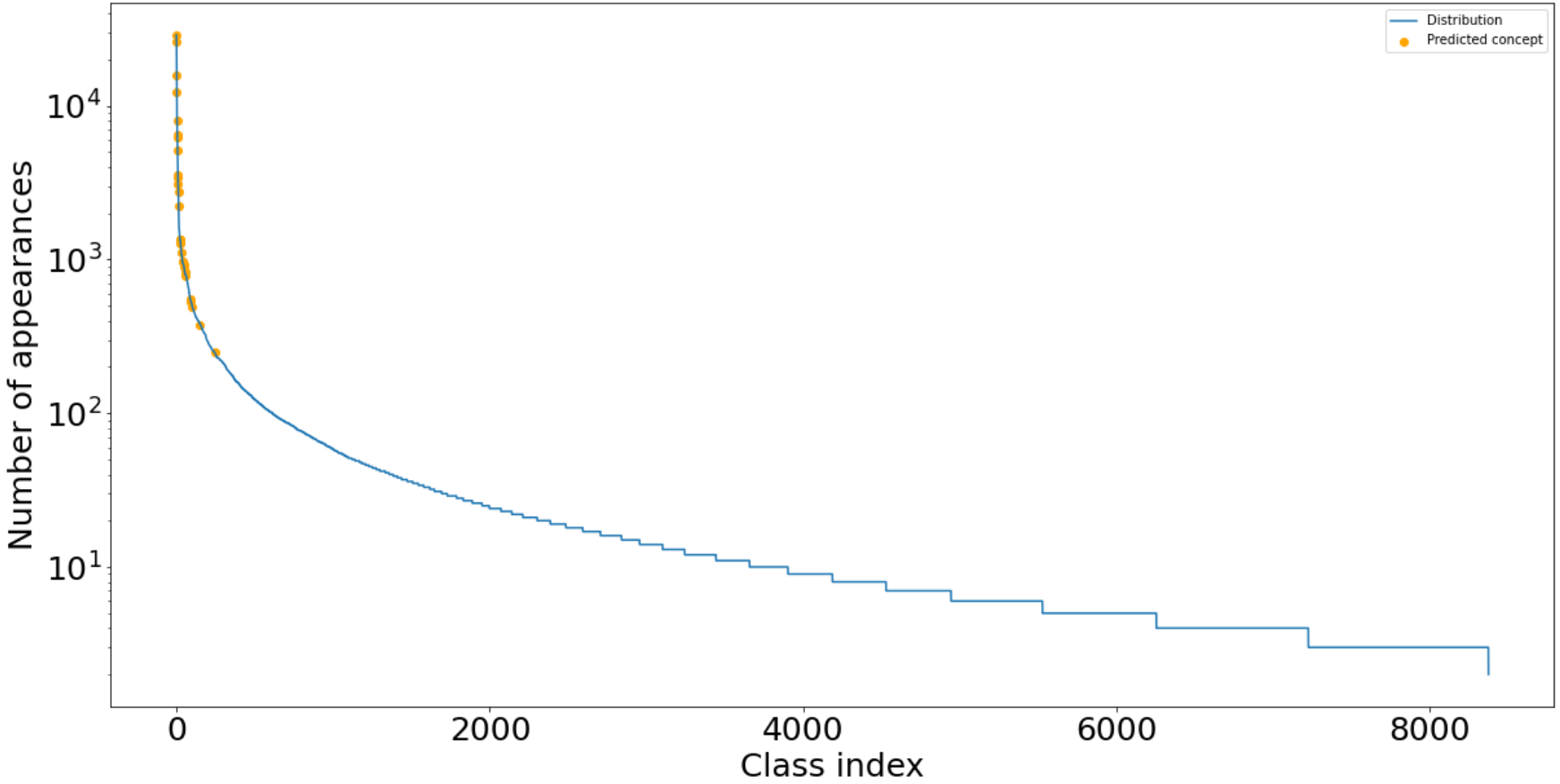
Figure taken from Z. Chen et al. "Multi-Label Image Recognition With Graph Convolutional Networks". In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). 2019.

Adding information from the labels – Results

- Current best score: $F_1^* = 45.85$ $(\mathcal{L}_{F_1} * \mathcal{L}_{BCE})$
- None of the experiments managed to surpass the previous baseline
- GCN improves the ablation test
- Interestingly, these models achieved these competitive scores by predicting a tiny portion of the label set (~30 labels)
  - This perhaps indicates that very few labels are responsible for the performance of all the models
  - Graph-based models managed to identify these crucial labels

Approach	F_1^* Score
ML-GCN (2-layer GCN - RANDOM INIT.)	45.56
ML-GCN (1-layer GCN - RANDOM INIT.)	**45.68**
ML-GCN (1-layer GCN - FASTTEXT INIT.)	45.02
ML-GCN (1-layer GCN - SENT2VEC INIT.)	45.35
ML-GCN (1-layer GCN - BERT INIT.)	44.78
ML-GCN (1-layer GCN - BERT INIT. (w/ projection))	45.45
Word Embeddings (FASTTEXT INIT.)	43.13
Word Embeddings (SENT2VEC INIT.)	44.88
Word Embeddings (BERT INIT.)	44.97

ML-GCN predictions



Teacher – Student self-training

- The goal is to provide extra examples for under-represented classes
- We used the ROCO[2] dataset as our unlabeled data
  - ROCO constitutes a superset of the competition's dataset

1: O. Pelka, S. Koitka, J. Rückert, F. Nensa, and C. M. Friedrich. "Radiology Objects in COntext (ROCO): A Multimodal Image Dataset". In: vol. 11043. Lecture Notes in Computer Science (LNCS), 2018, pp. 180–189.

Teacher – Student self-training (the algorithm)

Algorithm 1 Self-training (ST) for ToD

Input: Labeled data: L, Unlabeled data: U, Teacher: F^T, Student: F^S, Number of pseudo-labeled data in an iteration: k, Number of augmentations per input: q

Output: A trained Student F^S

Initialize F^T and train F^T on L
while F^S not good enough & $U \neq \emptyset$ **do**
  Initialize F^S, $L' \leftarrow Priority\_list()$
  for $x \in U$ **do**
    Compute prediction label $\hat{y}_x = F^T(x)$
    Compute confidence score s_x
    $L'.insert(\{x, \hat{y}_x, s_x\})$
  end for
  $L' \leftarrow L'.top(k)$
  $L \leftarrow L \cup L', U \leftarrow U \backslash L'$
  $L_{Aug} \leftarrow GradAug(L, F^T, q)$
  Train F^S on L_{Aug} with dropout
  $F^T \leftarrow F^S$
end while

Algorithm 2 Self-training with Teacher-Student (for specific labels)

Input: Labeled data: L, Unlabeled data: U, Teacher: F^T, Student: F^S, Number of pseudo-labeled data in an iteration: k, Labels to focus on: Q

Output: A trained Student model F^S

Initialize F^T and train F^T on L
while F^S not good enough & $U \neq \emptyset$ **do**
  Initialize F^S, $L' \leftarrow PriorityList()$
  for $x \in U$ **do**
    Compute prediction probabilities $\hat{y}_x = F^T(x) \in [0,1]^{|C|}$
  for $c \in Q$ **do**
    $L'.insert(\{\hat{y}_x[c], x\}) \quad \forall x \in U)$
    $L' \leftarrow L'.top(k)$ ▷ based on $\hat{y}_x[c]$
  $L \leftarrow L \cup L', U \leftarrow U \backslash L'$
  Train F^S ▷ noise can be added with Dropout and Augmentation
  $F^T \leftarrow F^S$

Algorithm adapted from F.Mi, W. Zhou, L. Kong, et al. "Self-training Improves Pre-training for Few-shot Learning in Task-oriented Dialog Systems". In: Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing. Online and Punta Cana, Dominican Republic, 2021, pp. 1887–1898.

Teacher – Student self-training (Results)

- We randomly picked 15 concepts, 5 from each ‘size band’ of the dataset
  - In essence, we picked 5 concepts that appeared frequently in our test set, 5 that appeared few times and 5 that appeared very few times
  - We ran the self-training algorithm for a fixed number of 2 iterations

Concept	Occurences	Per-class F_1 Score	
		Before	After
C0817096	1599	66.02	**68.31**
C0002978	1275	94.94	94.79
C0000726	1087	49.85	**50.26**
C0037303	848	86.38	**86.59**
C0221198	51	0.23	0.23
C0205126	79	0.0	**10.71**
C0046056	69	45.28	**48.74**
C3827002	56	14.00	**20.45**
C0449468	75	0.0	0.0
C0042149	18	0.0	**4.87**
C0025526	18	18.18	10.0
C0223683	16	11.11	10.0
C0225358	8	0.0	**9.09**
C0001209	10	0.0	0.0
C1629036	6	0.0	**50.0**

Self-supervised models

- We will focus on the experiments with CLIP[1] and Masked Autoencoder[2]

1: A. Radford et al. “Learning Transferable Visual Models From Natural Language Supervision”. In: Proceedings of the 38th International Conference on Machine Learning. Vol. 139. Proceedings of Machine Learning Research. Online, 2021, pp. 8748–8763.

2: K. He et al. “Masked Autoencoders Are Scalable Vision Learners”. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). New Orleans, LA, USA, 2022, pp. 16000–16009.

CLIP

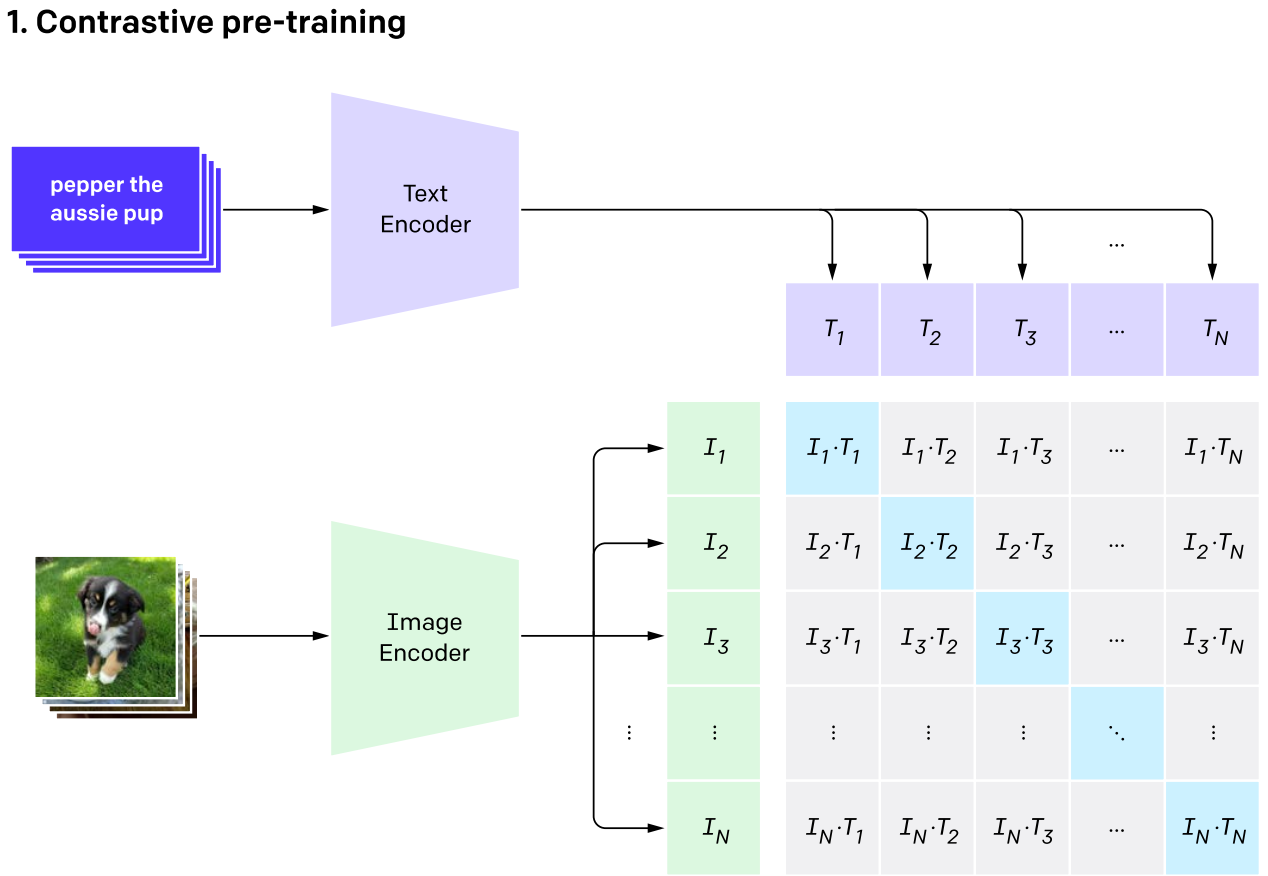


Figure taken from https://openai.com/blog/clip/

CLIP (cont.)

- The pre-trained ViT encoder of CLIP was used[1]
  - Extracted image embeddings using the last hidden state
- CLIP embeddings were concatenated with the embeddings extracted from the base image encoder acquiring $\tilde{v}$:

$$\tilde{\boldsymbol{v}} = [\boldsymbol{v}; \boldsymbol{v}_{CLIP}]$$

1: https://huggingface.co/openai/clip-vit-base-patch32

MAE

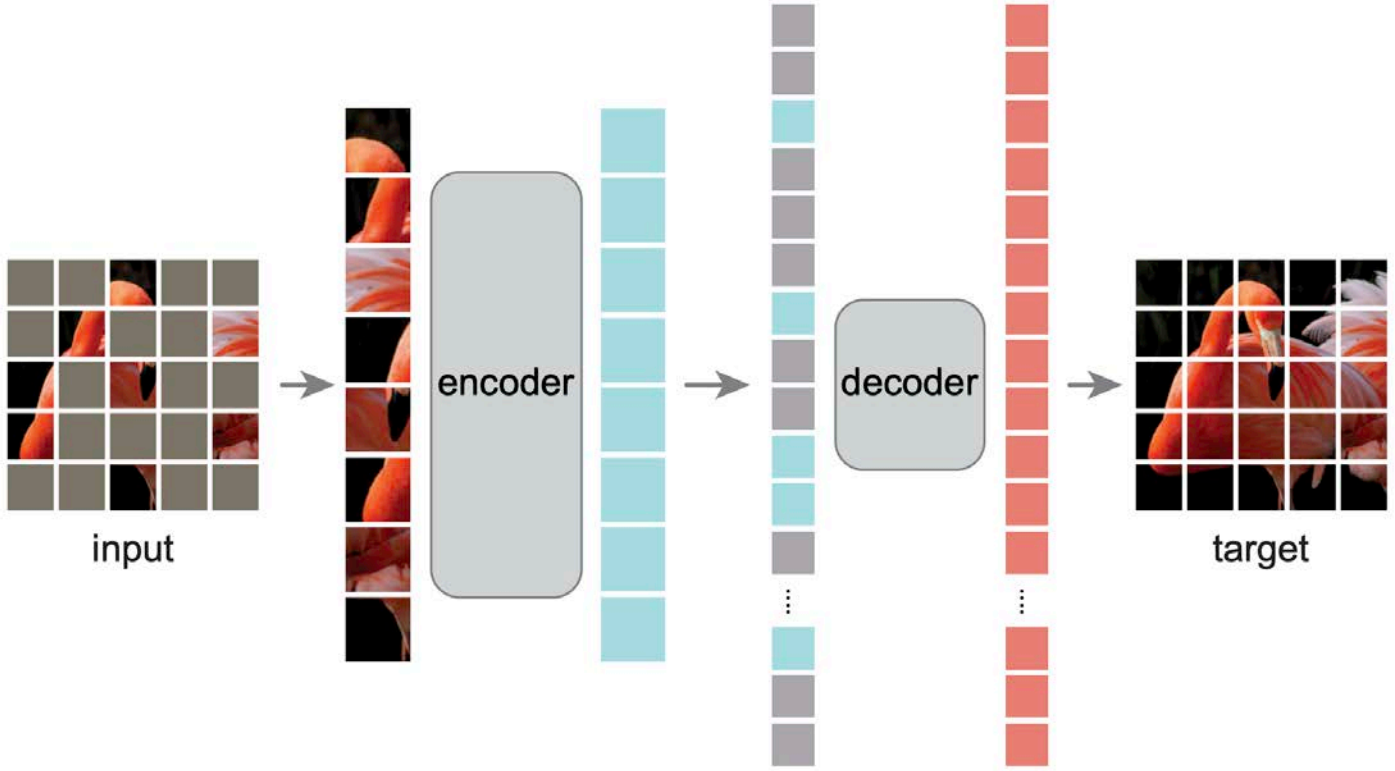


Figure taken from K. He et al. “Masked Autoencoders Are Scalable Vision Learners”. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). New Orleans, LA, USA, 2022, pp. 16000–16009.

MAE (cont.)

- Experimented with the ImageNet pre-trained model[1]
  - Linear probing + Fine-tuning
- Further pre-training on our dataset
  - Linear probing + Fine-tuning

1: https://huggingface.co/facebook/vit-mae-base

Self-supervised models – Results

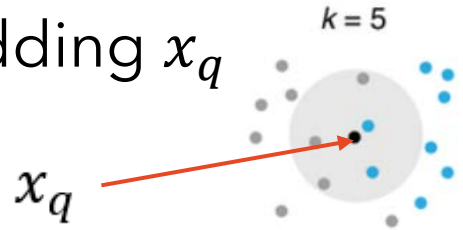
- Current best score: $F_1^* = 45.85$ $(\mathcal{L}_{F_1} * \mathcal{L}_{BCE})$
- MAE's pre-training improves the performance of the ViT backbone
  - Supervised ViT: $F_1^* = 39.86$
- CLIP showcases competitive performance without pre-training
  - Ablation test with Dropout shows that performance does not significantly drop when there is only signal from CLIP embeddings

Approach	F_1^* Score
MAE (Linear probing)	41.25
MAE (Fine-tuning)	43.58
MAE (pre-training + Linear probing)	43.27
MAE (pre-training + Fine-tuning)	44.49
CLIP (freezed embeddings)	**45.77**
CLIP (trainable)	45.68
CLIP + Dropout (base encoder signal)	45.42
CLIP + Dropout (both signals)	45.69

Neural Retrieval methods

k-Nearest Neighbors (k-NN)

- Based on the notion that similar patient conditions exhibit similar attributes depicted in the images
- Neural k-NN model
  - Similarity measured on image embeddings produced by a (trained) image encoder
  - Retrieve top k similar training images $\tilde{\boldsymbol{X}} = \{(x_{t_i}, y_i)\}_{i=1}^{k}$ according to the highest cosine similarity with the query embedding x_q



Weighted k-NN

- Weights vector $\boldsymbol{w} = [w_1, \ldots, w_k], \mathrm{w}_1 > \ \ldots > \mathrm{w}_\mathrm{k}$
  - 'Closer' neighbors have larger weights

Weighted k-NN – Voting scheme

- For each label l_i, calculate a weighted sum of k scores based on the neighbors of query x_q:[1]

$$f_i(x_q) = \frac{\sum_{j=1}^{k} w_j \cdot y_i\left(N_k(x_q, j)\right)}{\sum_{j=1}^{k} w_j}$$

presence of the i-th label on the j-th NN

- Monotonically decreasing weights: $w_1 = k, \ldots w_k = 1$

- l_i was assigned to x_q following: $h_i(x_q) = \begin{cases} 1, & f_i(x_q) \geq \tau \\ 0, & f_i(x_q) < \tau \end{cases}$

1:T. Chiang, H. Lo, and S. Lin. "A Ranking-based KNN Approach for Multi-Label Classification". In: Proceedings of the Asian Conference on Machine Learning. Vol. 25. Singapore Management University, Singapore, 2012, pp. 81–96

Weighted k-NN – Results

- k was tuned in $[5, 150]$ with step 5
  - Best value for τ was 0.4
- $F_1^* = 45.27,\ k = 10$

Retrieval-augmented classification

- Combine labels predicted from the retrieval component with labels predicted from the classifier

Retrieval-augmented classification – Interpolation

- Given x_q and the classifier's prediction vector $\hat{y}_m \in [0,1]^{|C|}$, x_q is also used to query the index of training image embeddings and obtain $\tilde{\boldsymbol{X}} = \{(x_{t_i}, y_i)\}_{i=1}^{k}$
- k-NN prediction is calculated as:[1]

$$\hat{y}_k = \sum_{i=1}^{k} \alpha_i y_i, \quad \alpha_i = \frac{e^{-d(x_{t_i}, \tilde{x})/\tau}}{\sum_j e^{-d(x_{t_j}, \tilde{x})/\tau}}$$

- The final prediction is calculated as the interpolation of $\hat{y}_m$ and $\hat{y}_k$:

$$\hat{y} = \lambda \hat{y}_k + (1-\lambda)\hat{y}_m, \quad \lambda \in [0,1]$$

1: X. Su et al. "Contrastive Learning-Enhanced Nearest Neighbor Mechanism for Multi-Label Text Classification". In: Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers). Dublin, Ireland, 2022, pp. 672–679.

Retrieval-augmented classification – Results

- Current best score: $F_1^* = 45.85$
- The interpolation approach managed to yield a very slight improvement when the inverted cosine similarity was used

Approach	Hyper-parameters	F_1^* Score
INTERPOLATION	$k = 10, \lambda = 0.5, \tau = 1$	42.44
INTERPOLATION	$k = 10, \lambda = 0.25, \tau = 1$	45.68
INTERPOLATION	$k = 5, \lambda = 0.25, \tau = 1$	45.77
INTERPOLATION	$k = 70, \lambda = 0.2, \tau = 1$	45.74
INTERPOLATION - INVERTED COS SIM.	$k = 5, \lambda = 0.25, \tau = 1$	**45.89**

ImageCLEFmedical Caption 2022 submissions

- We acquired the 1st position using an ensemble of two classifiers[1]
- All the additional methods and experiments were follow-up work

ID	*Run ID*	Approach	Primary F_1^*		Secondary F_1^*	Rank
			Development	Test		
cd1	182358	2xCNN[TL21]+FFNN@U	**47.01**	**45.11**	79.07	**1**
cd2	182356	2xCNN[TL21]+FFNN@I	46.32	44.27	**84.42**	12
cd3	182359	2xCNN[TL21]+FFNN/wk-NN	–	44.63	84.30	5
cd4	182340	CNN[TL21]+FFNN	45.71	44.39	81.20	9
cd5	182354	CNN[TL21]+FFNN(SAM+GC+$\mathcal{L}_{F_1}$)	46.06	45.02	82.14	3
cd6	182333	CNN[TL21]+wk-NN	45.25	43.05	84.40	28

1: F. Charalampakos et al. “AUEB NLP Group at ImageCLEFmedical Caption 2022”. In: CLEF2022 Working Notes. CEUR Workshop Proceedings. Bologna, Italy: CEUR-WS.org, 2022, pp. 1355-1373.

~~1. Introduction~~

~~2. Data~~

~~3. Methods & Results~~

4. Takeaways & Future work

Takeaways

- Imbalanced datasets consist a great challenge
  - In the multi-label scenario, oversampling is difficult
  - There is no panacea
- GNNs can be used when we want to exploit information about the importance of the labels
- More complex architectures are not necessarily better
  - ViTs are difficult to train
- Retrieval-based methods can be very competitive

Future Work

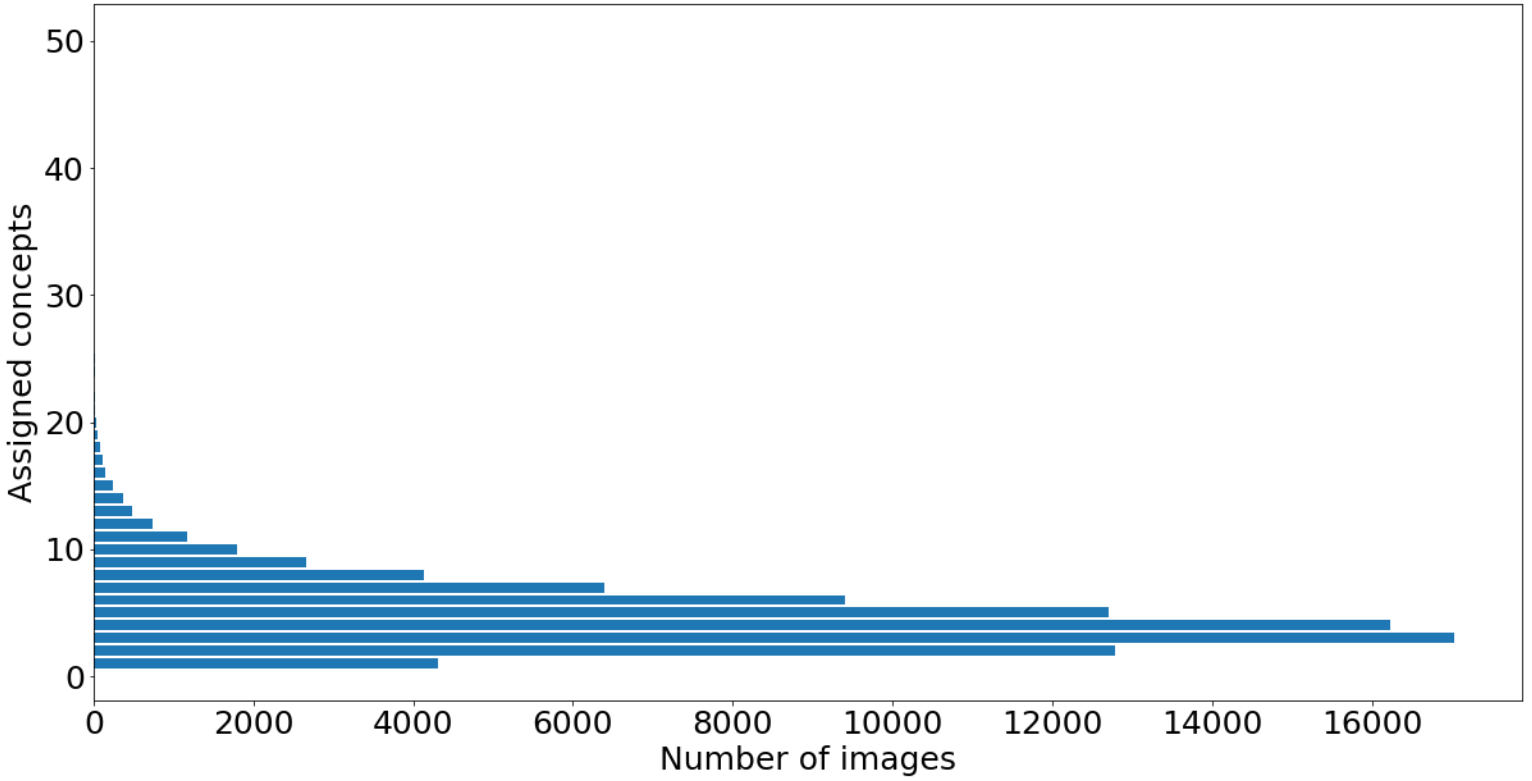
- Further experimentation with Teacher-Student self-training for tackling the imbalance
- Improve neural retrieval-based methods
  - Efficiency: create neighborhoods (e.g., clustering), eliminate unnecessary examples (e.g., genetic algorithms)
  - Evaluation performance: integrate a re-ranking step
- Integrate retrieval “augmentation” in the training phase
- Self-supervised pre-training (given sufficient data)
- Possible ways to pre-process medical images (e.g., proper augmentation)
- Explainability of the predictions

Thank you! ☺

Extra hidden slides

Concepts assignment distribution

- Average number of concepts assigned to each image: 4.74
- Minimum number: 1 (4,316 images)
- Maximum: 50 (1 image – outlier)



SAM loss function

$$\mathcal{L}^{SAM}(\mathbf{w}) = \max_{\|\epsilon\|_2 \leq \rho} \mathcal{L}(\mathbf{w} + \epsilon)$$

- $\boldsymbol{w} + \varepsilon$ is a perturbation of the model's parameters (can be roughly considered as the neighborhood)
- $\rho \geq 0$ is a hyper-parameter that denotes the size of the neighborhood
  - $\rho = 0.05$ is used
- $\mathcal{L}$ can be whatever loss function we choose

SAM loss function (cont.)

- Approximation of $\nabla \mathcal{L}^{SAM}(\mathbf{w})$:

$$\nabla \mathcal{L}^{SAM}(\mathbf{w}) \approx \nabla \mathcal{L}(\mathbf{w})|_{\mathbf{w}+\hat{\epsilon}(\mathbf{w})}$$

$$\hat{\epsilon}(\mathbf{w}) = \rho \text{sign}(g) \frac{|g|^{q-1}}{\left(\|g\|_q^q \right)^{\frac{1}{p}}}$$

- $p = q = 2$

Baseline: optimization – Gradient Centralization

- Centralizes the gradient vectors of weight matrices to have zero mean[1]
  - Acts as a regularization technique
- Given a weight vector $\boldsymbol{w_i}$ of a weight matrix $\boldsymbol{W}$:

$$\mathbf{\Phi}_{GC}(\nabla_{\mathbf{w}_i}\mathcal{L}) = \nabla_{\mathbf{w}_i}\mathcal{L} - \frac{1}{K}\sum_{j=1}^{K}\nabla_{\mathbf{w}_{i,j}}\mathcal{L}$$

1: H. Yong, J. Huang, X. Hua, and L. Zhang. "Gradient-Centralization: A New Optimization Technique for Deep Neural Networks". In: (2020), pp. 635–652.

Gradient Centralization – Math

- Equivalent matrix form:

$$\mathbf{\Phi}_{GC}(\nabla_{\mathbf{W}}\mathcal{L}) = \mathbf{P}\nabla_{\mathbf{W}}\mathcal{L}, \quad \mathbf{P} = \mathbf{I} - \mathbf{e}\mathbf{e}^{T}$$

- $\boldsymbol{P}$ is the projection matrix for the hyperplane with normal vector $\boldsymbol{e}$ in weight space determined by:

$$\mathbf{e}^{T}(\mathbf{w} - \mathbf{w}') = 0$$

- $\boldsymbol{P}\nabla_{\boldsymbol{W}}\mathcal{L}$ is the projected gradient
  - $\boldsymbol{W}$ is updated along its direction
- $\boldsymbol{e}^{\boldsymbol{T}}\boldsymbol{w}$ is constant during training ($\boldsymbol{e}^{T}\boldsymbol{w}^{t+1} = \ \ldots = \boldsymbol{e}^{T}\boldsymbol{w}^{0}$) from $\boldsymbol{e}^{T}(\boldsymbol{w} - \boldsymbol{w}') = 0$

Gradient Centralization – Math (cont.)

- This provides us with a constrained objective function (w.r.t to one weight vector $\boldsymbol{w}$) which regularizes the search space of $\boldsymbol{w}$:

$$\min_{\mathbf{w}} \mathcal{L}(\mathbf{w}), \quad \mathbf{e}^T(\mathbf{w} - \mathbf{w}^0) = 0$$

Baseline: optimization – GC (cont.)

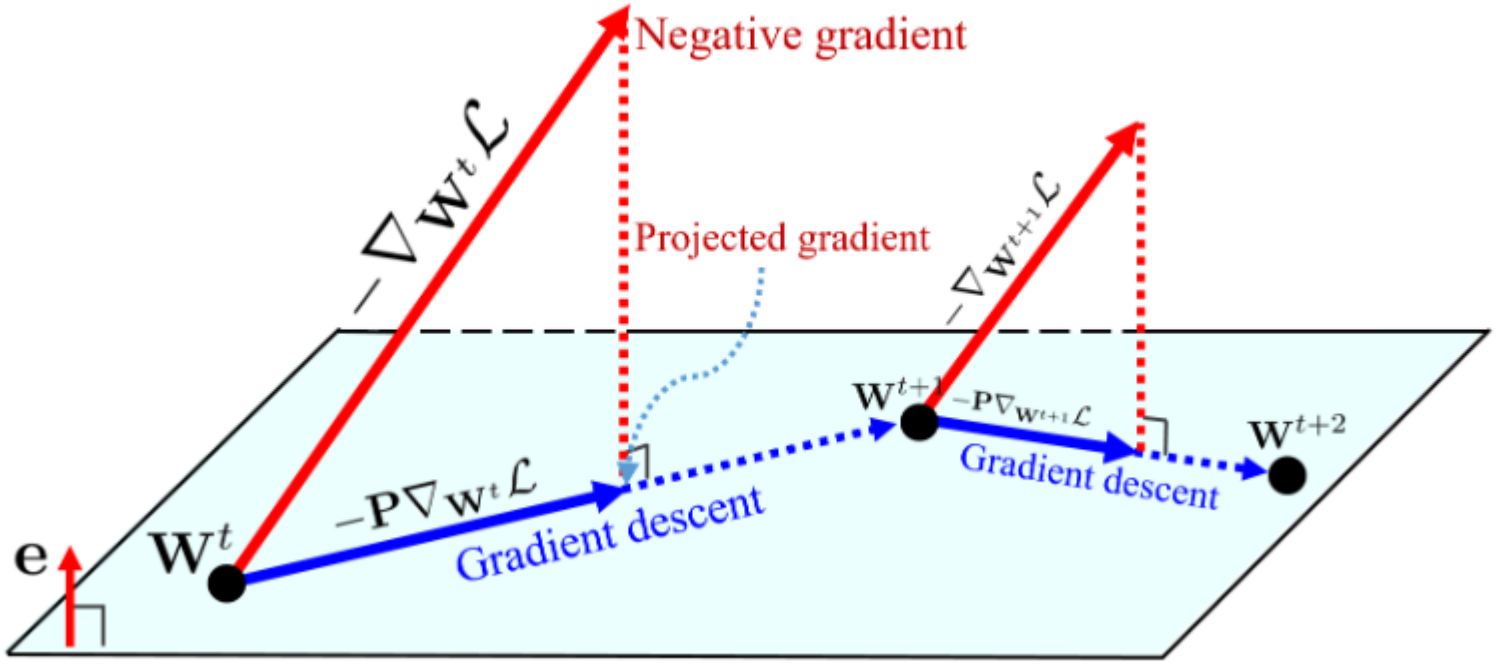


Figure taken from H. Yong, J. Huang, X. Hua, and L. Zhang. "Gradient-Centralization: A New Optimization Technique for Deep Neural Networks". In: (2020), pp. 635–652.

Bayesian Optimization – Insight

- BO is an iterative method that attempts to find the optimum of an (expensive) objective function f
- It incorporates prior belief about f (using a Gaussian Process – GP) and updates the prior with samples drawn from f to get a posterior that better approximates it
- As the number of samples grows with each iteration, the posterior distribution improves, and the algorithm becomes more certain of which regions in parameter space are worth exploring and which are not
- At each iteration, the GP is fitted to the known samples, and the posterior distribution, combined with an exploration strategy (e.g., Expected Improvement - EI), are used to determine the next point that should be explored

$$EI(x) = \mathbb{E}\max(\mathbf{f}(x) - \mathbf{f}(x^+), 0)$$

Baseline: optimization – Extended results

- SAM improves generalization by a small margin
- GC seems to hurt generalization in this task
  - Perhaps too strong regularization / constrained solution space
- BO did not manage to outperform the performance of the single threshold
  - Few iterations due to time restrictions
- BitFit[1] an BNFit[2] were ablations tests

Approach	F_1^* Score
$\mathcal{L}_{F_1}$ + SAM	**45.94**
$\mathcal{L}_{F_1}$ + GC	45.58
$\mathcal{L}_{F_1}$ + SAM + GC	45.75
$\mathcal{L}_{F_1}$ + BAYES OPT.	45.68
$\mathcal{L}_{F_1}$ + BitFit [ZGR22]	44.71
$\mathcal{L}_{F_1}$ + BNFit [FSM21]	44.95

1: E. Ben Zaken et al. "BitFit: Simple Parameter-efficient Fine-tuning for Transformer-based Masked Language-models". In: Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers). Dublin, Ireland, 2022, pp. 1–9
2: J. Frankle et al. "Training BatchNorm and Only BatchNorm: On the Expressive Power of Random Features in {CNN}s". In: 9th International Conference on Learning Representations, ICLR 2021. Vienna, Austria, 2021.

Ensembles

- Two set-based methods:

$$\text{UNION:}\ \hat{\mathcal{P}} = \mathcal{P}_1 \cup \ldots \cup \mathcal{P}_k$$

$$\text{INTERSECTION:}\ \hat{\mathcal{P}} = \mathcal{P}_1 \cap \ldots \cap \mathcal{P}_k$$

- Weights averaging using the Greedy Soup[1] mechanism

1: M. Wortsman et al. "Model soups: averaging weights of multiple fine-tuned models improves accuracy without increasing inference time". In: Proceedings of the 39th International Conference on Machine Learning. Vol. 162. Proceedings of Machine Learning Research. 2022, pp. 23965–23998.

Ensembles – Greedy Soup

Algorithm 1 GreedySoup

Input: Instances list $\theta = \{\theta_1, \ldots \theta_k\}$ (optionally sorted $\text{ValAcc}(\theta_i) \geq \text{ValAcc}(\theta_{i+1})$)

if θ is sorted **then**
    soup $\leftarrow \theta_1$
    $j^* \leftarrow 1$
else
    $j^* \leftarrow \arg\max\{\text{ValAcc}(\theta)\}$
    soup $\leftarrow \theta_{j^*}$
for $i \in \{1, \ldots, j^* - 1, j^* + 1, \ldots, k\}$ **do**
    if ValAcc(average(soup $\cup\{\theta_i\}$)) $\geq$ ValAcc(average(soup)) **then**
        soup $\leftarrow$ soup $\cup\{\theta_i\}$
return soup

Algorithm adapted from M. Wortsman et al. "Model soups: averaging weights of multiple fine-tuned models improves accuracy without increasing inference time". In: Proceedings of the 39th International Conference on Machine Learning. Vol. 162. Proceedings of Machine Learning Research. 2022, pp. 23965–23998.

Ensembles - Results

- The individual instances were trained using Monte Carlo cross-validation and a different random initialization
- More instances → worst score
- Data split plays an important role (more on that later)
- Validation score using model soups achieved higher performance
  - ~ 46.5 %

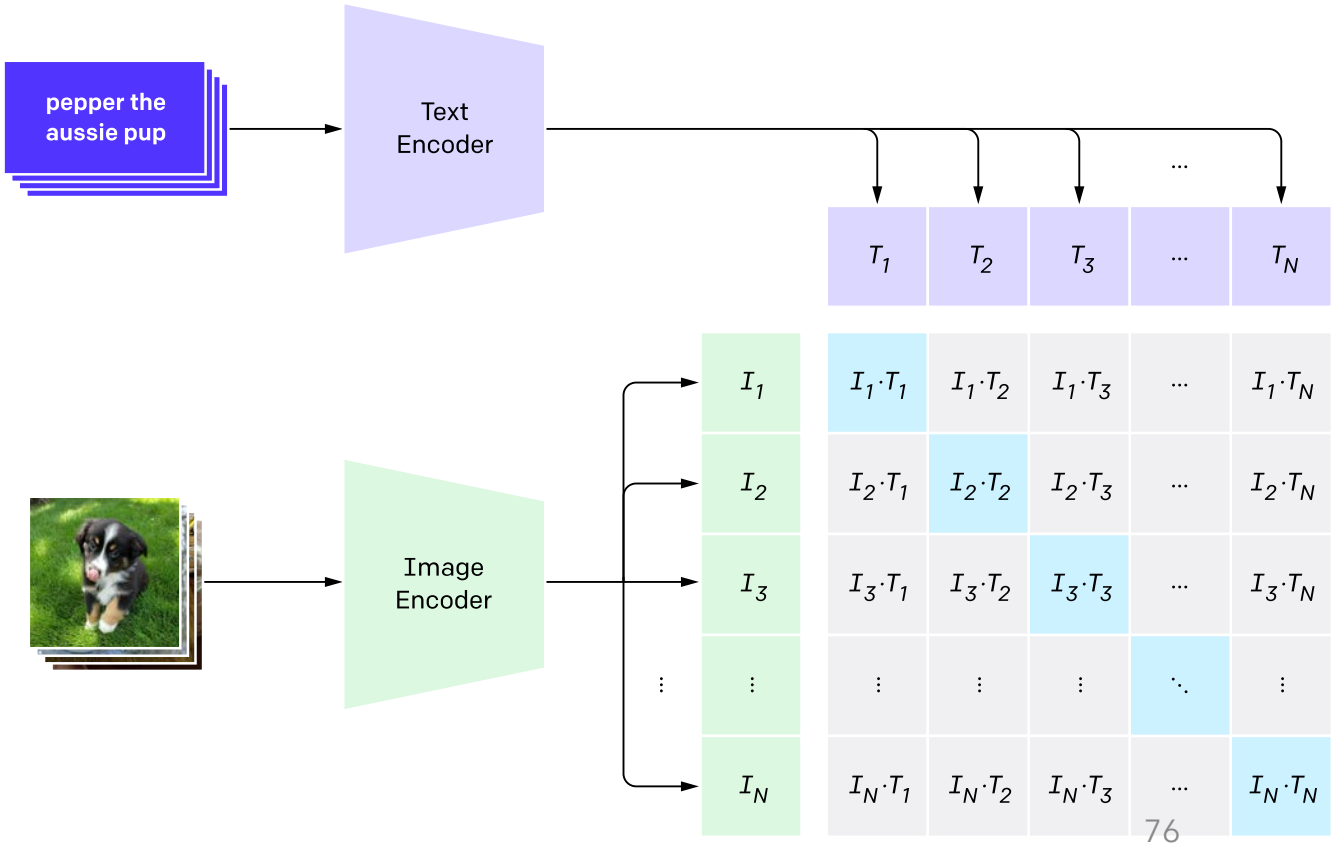
Approach	F_1^* Score
UNION - 2 instances	**45.91**
UNION - 3 instances	45.86
UNION - 4 instances	45.76
UNION - 5 instances	45.57
INTERSECTION - 2 instances	45.63
INTERSECTION - 3 instances	45.44
INTERSECTION - 4 instances	45.28
INTERSECTION - 5 instances	45.22
MODEL SOUPS	45.51

CLIP – Objective

- Consists of an image and text encoder
- It is trained using image/caption pairs
  - The goal is to map the matching pairs to nearby points on this joint space, and mismatching pairs to distant points in the embedding space
- It uses a contrastive objective between images and text embeddings

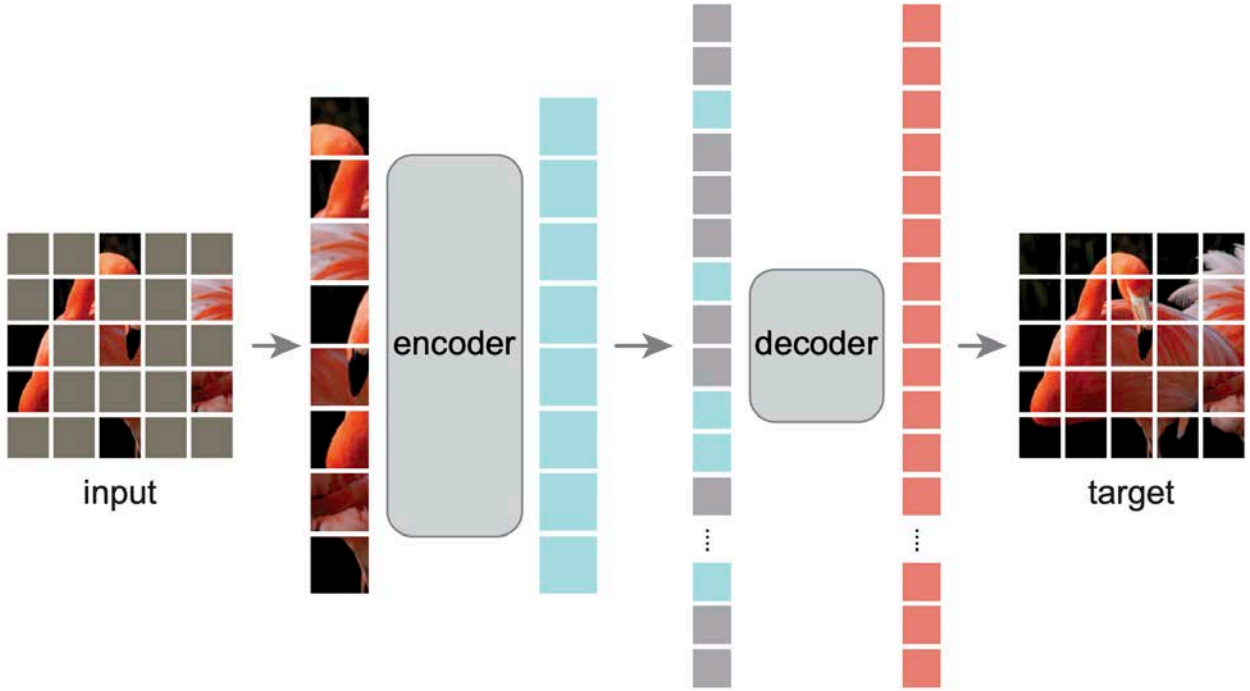
$$\mathcal{L}_{NCE}^{q} = -\log \frac{\exp(\langle q, k_+ \rangle / \tau)}{\sum_{j=0}^{N} \exp(\langle q, k_j \rangle / \tau)}$$

Figure taken from https://openai.com/blog/clip/



MAE – objective

- An autoencoder that masks random patches of the input image and aims at reconstructing the missing pixels
  - Similar to denoising autoencoders
- Encoder and Decoder ⟶ ViT architecture
  - Requires the input image to be divided into patches
- Mask a portion of the input patches.
  - The encoder is applied only to the visible (non-masked) patches
  - The decoder processes the full set of encoded and masked tokens
- MSE is used as an objective in the pixel space



dVAE (cont.)

- Experiments with DALLE's[1] and ViLMedic's dVAE[2] (pre-trained on MIMIC[3])
- Visual tokens were extracted using the codebook and concatenated with the embeddings extracted from the base image encoder acquiring $\tilde{v}$:

$$\tilde{\boldsymbol{v}} = [\boldsymbol{v}; \boldsymbol{v}_d]$$

1: https://github.com/openai/DALL-E
2: https://github.com/jbdel/vilmedic

3: A. E. W. Johnson et al. "MIMIC-CXR, a de-identified publicly available database of chest radiographs with free-text reports". In: Scientific Data 6 (2019), p. 317.

dVAE – Codebook

- Variational autoencoder that operates on a discrete latent space (posterior and prior distributions are categorical)
- A codebook with continuous embeddings is maintained
  - Used as a look-up table

$$z_q(x) = e_k, \quad k = \arg\max_j \left\| z_e(x) - e_j \right\|_2$$

- The corresponding embeddings of the codebook are then fed to the decoder

dVAE – Objective

- We want maximize the log-likelihood of the data distribution (evidence) $\log p_\theta(\boldsymbol{x})$

$$p_\theta(\mathbf{x}) = \int p_\theta(\mathbf{x}, \mathbf{z})\, dz = \int p_\theta(\mathbf{x}|\mathbf{z}) p_\theta(\mathbf{z})\, dz$$

intractable!

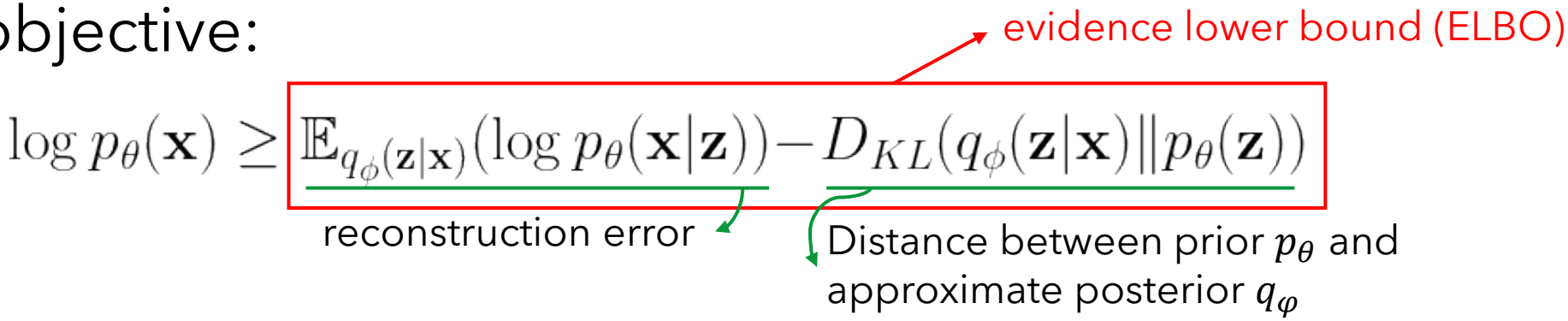
- Solution: approximate $p_\theta(\boldsymbol{x})$ with a "simpler" $q_\varphi(\boldsymbol{x})$
- Use Kullback-Leibler divergence[1] which measure the distance between two distributions as an extra objective

$$D_{KL} = (P\|Q) = \sum_i \log\left(\frac{P(i)}{Q(i)}\right)$$

1: S. Kullback and R. A. Leibler. "On information and sufficiency". In: The Annals of Mathematical Statistics 22.1 (1951), pp. 79–86.

dVAE – Objective (cont.)

- Final objective:



Self-supervised models – Results

- MAE's pre-training improves the performance of the ViT backbone
- CLIP showcases competitive performance without pre-training
  - Ablation test with Dropout shows that performance does not significantly drop when there is only signal from CLIP embeddings
- dVAE achieves the best performance
  - However, the Dropout ablation shows that the performance drops when only signal from the dVAE's tokens is passed through
  - Perhaps the objective does not allow learning of useful representations for classification

Approach	F_1^* Score
MAE (Linear probing)	41.25
MAE (Fine-tuning)	43.58
MAE (pre-training + Linear probing)	43.27
MAE (pre-training + Fine-tuning)	44.49
CLIP (freezed embeddings)	45.77
CLIP (trainable)	45.68
CLIP + Dropout (base encoder signal)	45.42
CLIP + Dropout (both signals)	45.69
dVAE (OpenAI)	45.74
dVAE (OpenAI + projection)	45.71
dVAE (OpenAI + projection + Dropout (base encoder signal))	25.99
dVAE (OpenAI + projection + Dropout (both signals))	**45.85**
dVAE (ViLMedic)	45.60
dVAE (ViLMedic + projection)	45.70
dVAE (ViLMedic + projection + Dropout (base encoder signal))	33.00
dVAE (ViLMedic + projection + Dropout (both signals))	45.62

Adding information from the labels – ML-GCN (adjacency matrix)

- Multi-Label Graph Convolutional Network (ML-GCN)
- Creation of adjacency matrix $\boldsymbol{A}$[1]
  - Initially create correlation matrix $\boldsymbol{M} \in R^{|C|\times|C|}$
    - $\boldsymbol{M}_{ij} \rightarrow$ co-occurrence of l_i, l_j
  - Acquire conditional probability matrix $\boldsymbol{P}, \boldsymbol{P_i} = \frac{\boldsymbol{M_i}}{\boldsymbol{N_i}}$
    - $\boldsymbol{N_i} \rightarrow$ occurrences of l_i
  - Binarize $\boldsymbol{P}$

$$\mathbf{A}_{ij} = \begin{cases} 0, & \mathbf{P}_{ij} < \tau \\ 1, & \mathbf{P}_{ij} \geq \tau \end{cases}$$

    - Binarization is done to filter out noisy edges (avoid over-fitting)

1: Z. Chen et al. "Multi-Label Image Recognition With Graph Convolutional Networks". In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). 2019.

Adding information from the labels – ML-GCN (adjacency matrix – cont.)

- A GCN is used in order to process $\boldsymbol{A}$

$$\mathbf{h}_{N(v)}^{k} = \mathrm{AGGREGATE}_k\left(\{\mathbf{h}_u^{k-1}, \forall u \in N(v)\}\right)$$

- When $AGGREGATE_k$ is the average operation:

$$H^{l+1} = f(H^l W_0^l + D^{-\frac{1}{2}} A D^{-\frac{1}{2}} H^l W_1^l)$$

- The feature of a node will be the weighted sum of its own feature and the adjacent nodes' features
- Use $\boldsymbol{A}'_{ij}$ to avoid over-smoothing problems:

$$\mathbf{A'}_{ij} = \begin{cases} (p/\sum_{\substack{i=1 \\ i\neq j}}^{C} \mathbf{A}_{ij}) \times \mathbf{A}_{ij}, & i \neq j \\ 1-p, & i = j \end{cases}$$

- p determines the weights assigned to a node itself and other correlated nodes
  - $p \to 1$: the feature of a node itself will not be considered, $p \to 0$: neighboring information tends to be ignored.

Adding information from the labels – ML-GCN (initial embeddings)

- For initial node embeddings, we experimented with random initialization, fastText pre-trained embeddings[1], sentence pre-trained embeddings[2] and BERT contextual embeddings[3]

1: Y. Zhang et al. “BioWordVec, improving biomedical word embeddings with subword information and MeSH”. In: Scientific Data 6.52 (2019).
2:Q. Chen et al. “BioSentVec: creating sentence embeddings for biomedical texts”. In: 2019 IEEE International Conference on Healthcare Informatics (ICHI). Los Alamitos, CA, USA, 2019, pp. 1–5.
3: M. Basaldella et al. “COMETA: A Corpus for Medical Entity Linking in the Social Media”. In: Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP). Online, 2020, pp. 3122–3137.

Adding information from the labels – LDGN

- Label-specific Dual Graph Neural network (LDGN)[1]
- Two step variation of ML-GCN
  - Initial node embeddings stem from a label-wise attention net that produces label-specific image embeddings $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_C$
  - Instead of using $\boldsymbol{A}'$, it reconstructs the adjacency matrix using a projection on $\boldsymbol{H}$ (output node embeddings):

$$\boldsymbol{A}^{\boldsymbol{R}} = \sigma\left(\left(F_{W_1}(\boldsymbol{H})\right)^T\left(F_{W_2}(\boldsymbol{H})\right)\right)$$

  - Re-learns the interactions among node embeddings with a second GCN now guided by $\boldsymbol{A}^{\boldsymbol{R}}$

1:Q. Ma, C. Yuan, W. Zhou, and S. Hu. "Label-Specific Dual Graph Neural Network for Multi-Label Text Classification". In: Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers). Online, 2021, pp. 3855–3864.

Adding information from the labels – LDGN (objective)

- The second GCN outputs node embeddings $\boldsymbol{H}' \in \mathbb{R}^{|C| \times D}$
- $\boldsymbol{H}$ and $\boldsymbol{H}'$ are concatenated and fed to a linear classifier to obtain probability scores over the label set:

$$\hat{y} = \sigma\left(F_\theta([\boldsymbol{H}; \boldsymbol{H}'])\right)$$

- The model can be trained end-to-end with a supervised loss

Adding information from the labels – DXML

- Deep embeddings for XML classification (DXML)[1]
- Dataset $\mathcal{D} = \{\boldsymbol{X}, \boldsymbol{Y}\} = \{x_i, y_i\}_{i=1}^{N}$, $x_i \in \mathbb{R}^D$ and $y_i \in [0,1]^{|C|}$
- Establish the label graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$
  - $\mathcal{V} = C$
  - There is an edge between two labels if they co-appear on at least one training example
- Given $\mathcal{G}$, an unsupervised node representation learning method is used to learn node embeddings through $\mathcal{G}'s$ structure
  - Node2Vec
  - GraphSAGE
  - Deep Graph Infomax (DGI)

1: W. Zhang, J. Yan, X. Wang, and H. Zha. “Deep Extreme Multi-Label Learning”. In: Proceedings of the 2018 ACM on International Conference on Multimedia Retrieval. New York, NY, USA: Association for Computing Machinery, 2018, pp. 100–107.

Adding information from the labels – DXML (objective)

- Project $\boldsymbol{Y}$ to

$$f_{\mathbf{Y}} = \{f_{\mathbf{y}_i} | f_{\mathbf{y}_i} = \frac{\mathbf{M}\mathbf{y}_i}{nnz(\mathbf{y}_i)}, \quad \forall \mathbf{y}_i \in \mathbf{Y}\}$$

- M is the learned embedding matrix
- Map $\boldsymbol{X}$ to $f_{\boldsymbol{X}} = F_\theta(\boldsymbol{X})$ by minimizing the distance between $f_{\boldsymbol{X}}$ and $f_{\boldsymbol{Y}}$
  - A distance loss was used as an objective
    - ℓ_1 (Huber), ℓ_2 (MSE) can be used

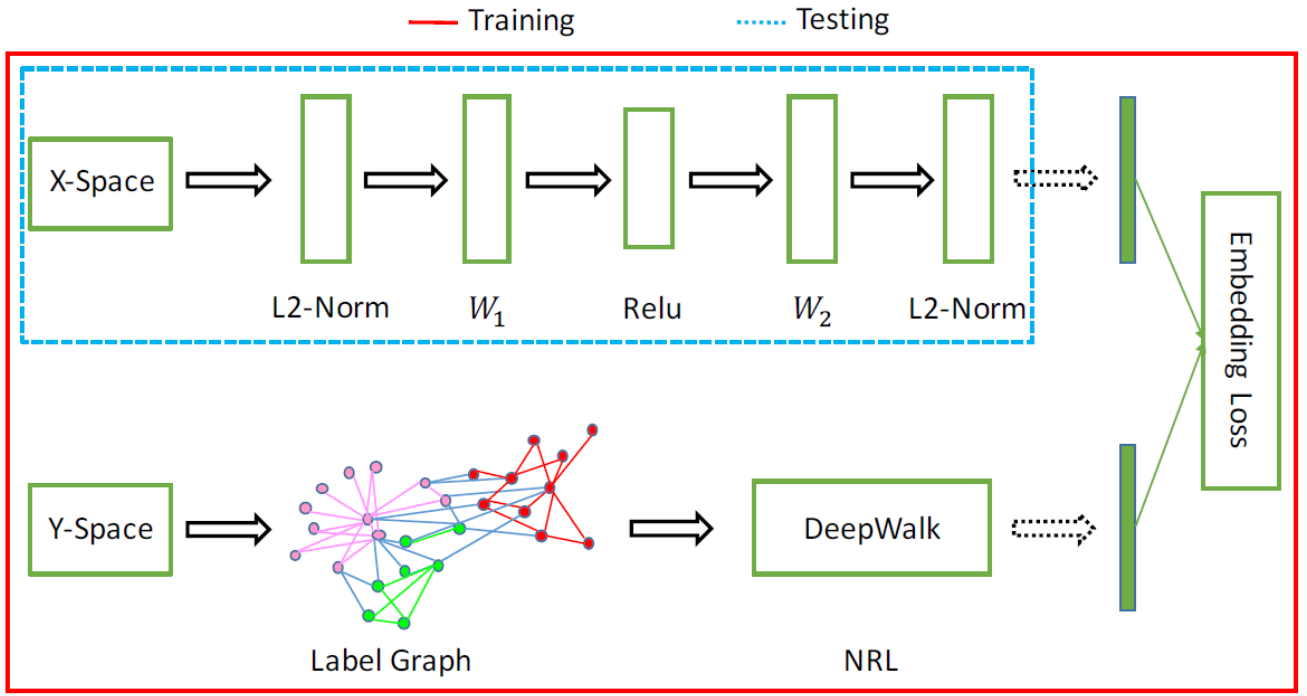


Figure taken from W. Zhang, J. Yan, X. Wang, and H. Zha. "Deep Extreme Multi-Label Learning". In: Proceedings of the 2018 ACM on International Conference on Multimedia Retrieval. New York, NY, USA: Association for Computing Machinery, 2018, pp. 100–107.

Node2Vec

- Node2Vec:[1]

$$\mathcal{L}_{N2V} = -\sum_{u \in V} \sum_{v \in N_R(u)} \log(p(v|u)) = -\sum_{u \in V} \sum_{v \in N_R(u)} \log\left(\frac{\exp(z_u^T z_v)}{\sum_{n \in V} \exp(z_u^T z_n)}\right)$$

- Optimize embeddings to maximize the likelihood of random walk co-occurrences (like Word2Vec's SKIP-GRAM model)
- Random walks are biased, trade off between local (BFS) and global (DFS) views of the graph
- Node2Vec creates a look-up embedding table (like Word2Vec)

1: A. Grover and J. Leskovec. "node2vec: Scalable Feature Learning for Networks". In: Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and Data Mining. San Francisco, CA, USA, 2016, pp. 855–864.

GraphSAGE

- GraphSAGE[1] advances the idea of Node2Vec by replacing the look-up table with a deep graph encoder (e.g., a Graph Convolutional Network – GCN[2]) providing inductive representation learning (i.e., embedding calculation for unseen nodes)

$$\mathcal{L}_{GS}(z_u) = -\log(\sigma(z_u^T z_v)) - Q\mathbb{E}_{v_n \sim P_n(v)} \log(\sigma(-z_u^T z_{v_n})), \quad \forall u \in V$$

- The random walk-based objective encourages nearby nodes to have similar representations, while enforcing the representations of disparate nodes to be distinct

1: W. Hamilton, Z. Ying, and J. Leskovec. “Inductive Representation Learning on Large Graphs”. In: Advances in Neural Information Processing Systems. Long Beach, CA, USA, 2017.

2: T. N. Kipf and M. Welling. “Semi-Supervised Classification with Graph Convolutional Networks”. In: 5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings. Toulon, France, 2017

DGI

- Aims at maximizing the mutual information between patch representations[1]
  - patch → neighborhood
  - Encoded by a E (e.g., a GCN) in $\boldsymbol{h}_u$ around a node u
- Train E in order to obtain local representations that capture the global information content of the entire graph
  - Graph is represented by $\boldsymbol{s} = \mathcal{R}(\{\boldsymbol{h}_u | u \in \mathcal{G}\})$. $\mathcal{R}$ can be e.g., mean

1: P. Velickovic, W. Fedus, W. L. Hamilton, et al. “Deep Graph Infomax”. In: 7th International Conference on Learning Representations, ICLR. New Orleans, LA, USA, 2019.

DGI objective

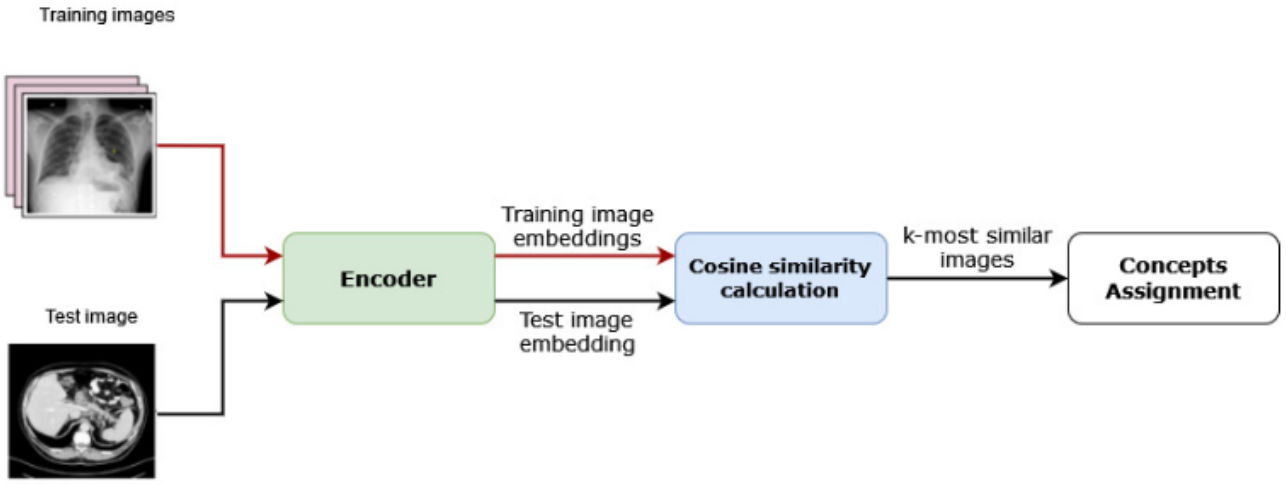
- Maximize mutual information between $\boldsymbol{h}_u$ and $\boldsymbol{s}$

$$\mathcal{L}_D = \frac{1}{N+M}\left(\sum_{i=1}^{N}\mathbb{E}_{\mathcal{G}}\left[\log\left(\mathcal{D}\left(\mathbf{h}_u,\mathbf{s}\right)\right)\right]+\sum_{j=1}^{M}\mathbb{E}_{\hat{\mathcal{G}}}\left[\log\left(1-\mathcal{D}\left(\hat{\mathbf{h}}_u,\mathbf{s}\right)\right)\right]\right)$$

- $\mathcal{D}$ is a similarity measure (e.g., cosine)
- $\hat{\mathcal{G}} = \mathcal{C}(\mathcal{G})$, $\mathcal{C}$ is a corruption function

Unweighted k-NN – Label assignment

- Assigned the r most frequent concepts present in the k retrieved images (essentially majority voting)
- We tuned k and r using the validation set.
  - For k, we experimented with values between 1 and 200
  - For r, we used values between 1 and 10 and also considered the average number of labels present in the k retrieved images



Encoder	Hyper-parameters	F_1^* Score
EfficientNetV2B0[TL21]-supervised	$k = 73, r = 2$	43.78

Weighted k-NN – 2nd voting scheme

- The weight of each neighbor is now a function of its similarity with test image x_q:[1]

$$w_j = \frac{d_1+\epsilon}{d_j+\epsilon}$$

- d is a distance metric (e.g., cosine, inverted similarity etc.)
- For each label l_i:

$$v_i = \sum_{j=1}^{k} y_i(N_k(x_q, j))w_j$$

1:A. Kumar, S. Dyer, J. Kim, et al. “Adapting content-based image retrieval techniques for the semantic annotation of medical images”. In: Computerized Medical Imaging and Graphics 49 (2016), pp. 37–45

Weighted k-NN – Results

- For the 1st voting scheme, k was tuned in $[5, 150]$ with step 5
  - Best value for τ was 0.4
- For the 2nd scheme, k was tuned in $[5, 100]$ with step 10
  - The r highest voted labels were picked. r was fixed at 2
  - Cosine distance was used as the metric

Approach	Hyper-parameters	F_1^* Score
Linear weighting (EfficientNetV2B0[TL21]-supervised)	$k = 10$	**45.27**
Distance weighting (EfficientNetV2B0[TL21]-supervised)	$k = 60, r = 2$	43.97

Retrieval-augmented classification – naïve approach

- The k-NN system was used in order to add (set union) its predicted labels to the classifier's predicted label set
  - r most frequent labels amongst the k neighbors were selected

Retrieval-augmented classification – Interpolation (contrastive objective)

- In order to improve the quality of retrieved neighbors a contrastive learning objective to train the encoder was proposed:

$$\mathcal{L}_{con}^{ij} = -\beta_{ij} \log \frac{e^{-d(z_i, z_j)/\tau'}}{\sum_{k \in g(i)} e^{-d(z_i, z_k)/\tau'}} \qquad \beta_{ij} = \frac{C_{ij}}{\sum_{k \in g(i)} C_{ik}}, \quad C_{ij} = y_i^T y_j$$

$g(i) = \{j \mid j \in \{1, \ldots, b\}, j \neq i\}$

$$\mathcal{L}_{con} = \sum_i \sum_{j \in g(i)} \mathcal{L}_{con}^{ij} \qquad \mathcal{L} = \mathcal{L}_{BCE} + \gamma \mathcal{L}_{con}$$

Retrieval-augmented classification – Extended results

- The naïve approach caused a drop of performance
  - Slightly better recall, much lower precision
- UNION(RARE) considered only very rare labels (from the tail of the distribution
  - ≤ 20 occurrences
- The interpolation approach managed to yield a very slight improvement when the inverted cosine similarity was used
- The contrastive objective did not offer any benefits in this task
  - More extensive tuning needed (?)

Approach	Hyper-parameters	F_1^* Score
UNION	$k = 10, r = 2$	34.66
UNION	$k = 10, r = 1$	39.02
UNION	$k = 15, r = 1$	40.36
UNION	$k = 73, r = 1$	40.21
UNION (RARE)	$k = 15, r = 1$	37.81
INTERPOLATION	$k = 10, \lambda = 0.5, \tau = 1$	42.44
INTERPOLATION	$k = 10, \lambda = 0.25, \tau = 1$	45.68
INTERPOLATION	$k = 5, \lambda = 0.25, \tau = 1$	45.77
INTERPOLATION	$k = 70, \lambda = 0.2, \tau = 1$	45.74
INTERPOLATION - INVERTED COS SIM.	$k = 5, \lambda = 0.25, \tau = 1$	**45.89**
INTERPOLATION + CL [SWD22]	$k = 5, \lambda = 0.25, \tau = 1, \gamma = 0.005, \tau' = 10$	45.74