Department of Computer Science and Engineering
Indian Institute of Technology, Jodhpur

Semester IV
Academic Year 2019-20
B.Tech. Project Report

Multi-Label Image Annotation using Image and Label Graphs

Manul Goyal (B18CSE031)
Manan Shah (B18CSE030)

Supervisor: Dr. Yashaswi Verma

Abstract. *The objective of our project is to compare the exhaustive and diversification powers of various models suggested by different authors to solve the multilabel image classification problem. We also attempt to incorporate the semantic weights from the Diverse Image Annotation (DIA)*[1] *model into the ML-GCN*[10] *model's adjacency matrix to improve its diversification ability, and extend the single label binary classification model to multilabel classification. The four major models that we are comparing are the baseline algorithm*[5]*, ML-GCN*[10]*, ML-GCN with semantic modifications, and multilabel classification using GCN-based binary classifiers, which we call BIN-GCN. The baseline model uses the k-nearest neighbour approach to identify the images most similar to the test image (based on some selected features) and assign labels greedily to the test image. We get the model's performance on the IAPR TC-12 dataset to establish a reference for comparing the other models with. Then we apply the ML-GCN model on the IAPR TC-12 dataset, which generates classifiers for each label using a Graph Convolutional Network (GCN) and applies them on the features extracted from the test image by a pretrained CNN model (ResNet-101). We also modify the adjacency matrix for the GCN using the semantic weights of the labels obtained from DIA, so as to improve its diversification abilities. Finally, we extend an existing approach for single-label binary classification to multilabel classification using separate models for each label and training each model on an appropriate set of positive and negative images. We attempt negative sample mining using a stratification algorithm*[8] *for preparing the training sets for these models. We compare two approaches for picking the final k labels from the resulting label scores in BIN-GCN and ML-GCN*[10]*. Thus, in ML-GCN*[10]*, a graph is constructed among labels, while in BIN-GCN, a graph is constructed among images, and then GCN is applied on the respective graphs, capturing the dependencies between labels in ML-GCN and between images in BIN-GCN. We compare the two approaches.*

Contents

1. Introduction

Multilabel image classification is a practical task which involves assigning some labels from a given dictionary of labels to a given test image, which best describe the test image. Usually, there is an upper limit on the number of labels assigned to each image, and thus the task has two main objectives. The first obvious objective is to cover as many aspects of the image as possible, which includes covering as many objects and actions depicted in the image as possible using a limited number of labels. The second objective is to ensure diversity between the labels assigned to an image, to remove redundancy and improve coverage. Since there is an upper limit on the number of labels to be assigned, fulfilling the second objective of diversity helps to fulfill the first objective of exhaustiveness. To measure the exhaustiveness and correctness of the model's predictions, we compute the average example-based precision (EP), recall (ER) and F_1 score (EF1), and the average class-based precision (CP), recall (CR) and F_1 score (CF1). To measure the diversity in the labels predicted by the model, we compute some semantic metrics, visually, the semantic precision (SP), semantic recall (SR) and semantic F_1 score (SF1). The last three semantic metrics are taken from [1]. These nine metrics are explained in section 3.

2. Models

In this section we give an overview of the various models mentioned above as well as explain the basis of our modifications to them.

2.1. Baseline Approach

The baseline model proposed in [5] is based on the hypothesis that images which have similar appearance are likely to share labels. For annotation of an image, it finds nearest images from the train set and then transfer labels greedily. The distance between images are calculated using low-level image features. We report the results of this approach on two datasets, Corel5k and IAPR TC-12. The feature used for both are mentioned in Table 1.

2.1.1. Distance Calculation Among Images

An image can be represented with the help of various features. Let us say we have an image I_i with features $f_i^1, f_i^2, ..., f_i^N$ (features are normalized before use). The distance between images will be the combination of the distances between various features taken into consideration. To determine the corresponding distances, there are four common distance measures used for histograms and distributions (KL-divergence, χ^2 statistic, L_1 distance and L_2 distance) as mentioned in [5]. Different distance measures are used for different features. After calculating feature distances ($(d(i,j)^k$ as the distance between f_i^k and f_j^k calculated with the use of suitable distance measures), the next part is determining how much each feature distance $d(i,j)^k$ must contribute to the total distance $d(i,j)$ between the images i and j. We have used the **JEC (Joint Equal Contribution)** scheme proposed in [5], where each feature is given equal weightage. However, we need to scale each $d(i,j)^k$ with appropriate scaling terms, since different features have different ranges. Each feature distance is scaled based on the minimum and the maximum distance corresponding to this feature in the train set (scaled distance between the k^{th} feature, $d(i,j)_n^k = (d(i,j)^k - min_{i,j} d(i,j)^k)/(max_{i,j} d(i,j)^k - min_{i,j} d(i,j)^k)$).For IAPR TC-12, since we are using only one feature, this step is unnecessary, so $d(i,j)$ = $d(i,j)^1$ directly. For Corel5k, $d(i,j)$ = $\sum_{k=1}^{N} d(i,j)_n^k/N$ (where $d(i,j)_n^k$ is scaled distance). Table 1 shows the various features and corresponding distances we have used for the Corel5k and IAPR TC-12 datasets.

Dataset	Feature Name	Dim.	Dist.
$Corel5k$[9]	(1) Dense SIFT, (2) Harris SIFT	1000	χ^2
	(3) Dense SIFT (V3H1), (4) Harris SIFT (V3H1)	3000	χ^2
	(5) GIST	512	L_2
	(6) Dense Hue, (7) Harris Hue	100	χ^2
	(8) Dense Hue (V3H1), (9) Harris Hue (V3H1)	300	χ^2
	(10) RGB, (11) HSV, (12) LAB	4096	L_1
	(13) RGB (V3H1), (14) HSV (V3H1), (15) LAB (V3H1)	5184	L_1
$IAPRTC-12$[1]	VGGF-PCA (from DIA [])	536	L_2

Table 1. Features and corresponding distances used for IAPR TC-12 and Corel5k

2.1.2. Label Transfer

A greedy algorithm mentioned in [5] has been implemented by us. Consider for a test image I_T, we are required to assign n labels from K- *nearest neighbours* of test image in training set. Let $I_1, I_2, ...I_K$ be the K nearest train images for a given test image is ascending order of distances. The labels are transferred according to the below mentioned steps :

1. List the labels of I_1 according to there frequency in train dataset and transfer all labels if $n < |I_1|$ or else transfer only the initial n labels of the list and end label transfer.
2. List the labels from I_2 to I_K according to two factors : (1) Co-occurrence with the labels transferred from I_1 in training set (2) Local frequency (occurrence of labels in set I_2 to I_K).

We also tried to modify the greedy algorithm by combining the factors mentioned in step 2 in various manners. We also report that reaults for Corel5k and IAPR TC-12.

2.2. Multi-Label Image Recognition with GCN (ML-GCN)[10]

The overall model for Multi-Label Image Recognition using GCN is shown in Figure 1. We report the results of this approach on two datasets, Pascal VOC and IAPR TC-12.

2.2.1. Graph Convolution Network (GCN)

Graph Convolution Network is a Neural Network that operates on a graph. Given a graph G = (V,E) then input for GCN are a feature matrix X of dimension $N \times F^0$ and an adjacency matrix A of dimension $N \times N$, where N is the number of nodes and F^0 is the size of input feature. Then a simple propagation rule for any layer in GCN can be given by : $\mathbf{f}(\mathbf{H^{i+1}}, \mathbf{A}) = \sigma(\mathbf{AH^iW^i})$ as mentioned in [3], where A is an adjacency matrix of dimension $N \times N$, H^i is a node feature matrix, input for i^{th} layer of dimension $N \times F^i$,W^i is a transformation matrix to be learned of dimension $F^i \times F^{i+1}$, $\sigma()$ is a non-linear activation function such as ReLU function and H^{i+1} is a node feature matrix, output of i^{th} layer of dimension $N \times F^{i+1}$. After propagation the representation of each node is sum of its neighbors features. So, each layer of GCN represents each node as aggregation of its neighborhood.

The problems mentioned in [3] in above propagation are : (1) The layer of GCN does not take into account the nodes feature and (2) Nodes with large degrees will have large value in their feature representation but nodes with small degrees will have smaller value which can cause vanishing or exploding gradients. The first problem can be solved by adding identity matrix I to adjacency matrix A. The second problem of normalizing the feature representations can be solved by

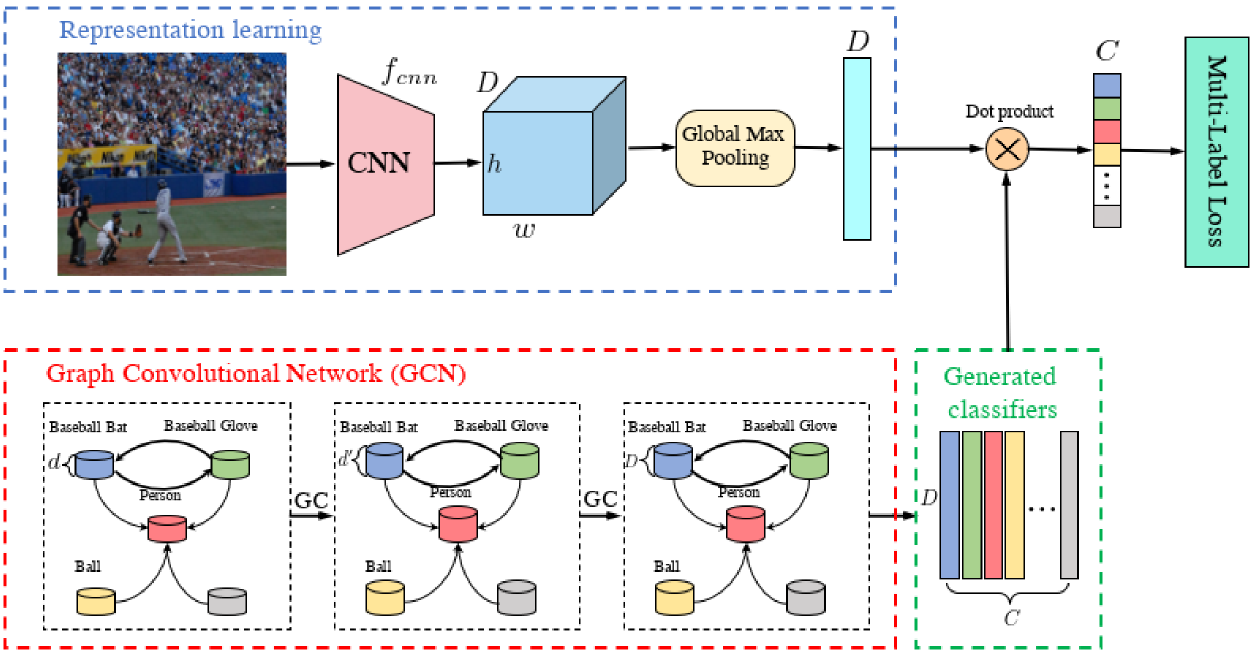

Figure 1. Input image features X are being extracted using convolution neural network and pooling layer. Word embeddings (300-dim GloVe) are used as input node features for GCN. From stacked GCNs a set of inter-dependent object classifiers W is obtained. Labels scores are predicted using X and W. Figure taken from [10].

normalizing our adjacency matrix, in such a way that weights in each row of adjacency matrix have been divided by the degree of that node.

For annotation tasks we are using the GCN layer, represented as $f(H^{i+1}, A) = \sigma(\hat{A}H^iW^i)$, where $\hat{A}$ is our normalized correlation matrix and the non-linear function is LeakyReLU, as mentined in [10]. So we can learn and model the inter-relationship of among labels.

2.2.2. Model for Multi-Label Image Recognition

The model discribed in [10] is based on two models (basically two builtin modules), first one is image representation learning and another is GCN based classifier learning.

Image Representation Learning : It takes images and learn about its features. Any CNN base model can be used, we have used ResNet-101 as mentioned in [10]. The input for the model will be an image of dimension 448×448 and output of $2048 \times 14 \times 14$ dimension is obtained from ResNet-101 and Global Max-Pooling is applied to get our feature representation matrix $\mathbf{X}$ of dimension 2048 for that image.

GCN Based Classifier Learning : For our image annotation task we don't have any predefined correlation matrix (i.e. adjacency matrix for GCN), how it is built is explained in further section. The input feature matrix for our GCN will be word embeddings of labels, which are obtained from the 300-dimensional GloVe trained on the Wikipedia dataset[7]. The GCN will map these label embeddings into a set of inter-dependent classifiers, which will be further applied to $\mathbf{X}$. As word embeddings to classifier mapping is shared across all classes, the classifier obtained from stacked GCNs can retain weak semantic structures in word embedding space. Here, the tranformation matrix is being learned to obtain learned classifier. For first layer the input will be a matrix $\mathbf{Z}$ of dimension $C \times d$, where C is the number of labels and d is the dimension of word embeddings for each label. The final output of our stacked GCNs will be a matrix $\mathbf{W}$ of dimension $C \times D$, where D is the dimension of image feature representation. The prediction score for each image $\hat{\mathbf{y}}$ can be obtained by combination of learned classifier and image representation as

$$\hat{\mathbf{y}} = \mathbf{W}\mathbf{X} \tag{1}$$

The ground truth labels of an image ($\mathbf{y}$ of dimension C) contains 0 and 1 to indicate whether that particular label appears or not. The network obtained by the combination of above two models is trained with traditional multi-label classification loss as mentioned in [10] :

$$L = \sum_{c=1}^{C} y^c log(\sigma(\hat{y}^c)) + (1 - y^c) log(1 - \sigma(\hat{y}^c)) \tag{2}$$

2.2.3. Correlation Matrix for ML-GCN

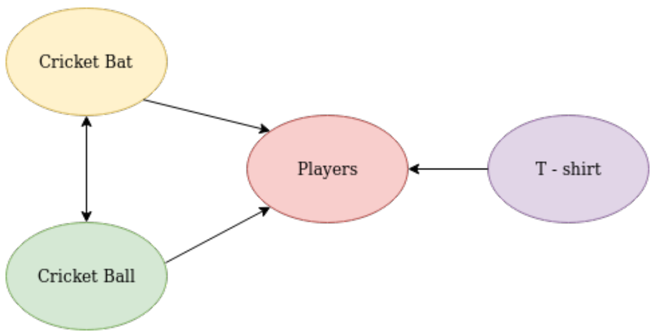

Figure 2. The directed edge between label A and label B shows that if label A is present then label B is likely to appear, but reverse is not true. Correlation matrix must be created appropriately to show this behaviour.

Our GCN layer uses correlation matrix for propagation of information between nodes and here there is no predefined correlation matrix. The matrix must be built, to show the behaviour as shown in Figure 2. It is built in the form of conditional probability, i.e. $P_{ij} = P(L_j|L_i)$ means probability of occurrence of label j when label i appears as mentioned in [10]. Also correlation matrix is asymmetrical because of inequality of $P(L_i|L_j)$ and $P(L_j|L_i)$. Let $\mathbf{M}$ be a co-occurrence matrix where M_{ij} denotes the co-occurrence of label i and label j. So each row of conditional probability matrix can be given by $\mathbf{P_i} = \mathbf{M_i}/N_i$, where N_i is the total occurrence of label i in training set. Co-relation matrix created in such a way may have noise and can be over-fitted to the training set. So a binary correlation matrix $\mathbf{A}$ is created with the use of a threshold τ.

$$A_{ij} = \begin{cases} 1, & \text{if } P_{ij} \geq \tau \\ 0, & \text{if } P_{ij} < \tau \end{cases} \tag{3}$$

Because of Binary Correlation matrix it may lead to Over-Smoothing problem as mentioned in [10], i.e. labels from different clusters may become interconnected. Hence the matrix is re-weighted as

$$\hat{A}_{ij} = \begin{cases} p / \sum_{j=1}^{C} A_{ij}, & \text{if } i \neq j \\ 1 - p, & \text{if } i = j \end{cases} \tag{4}$$

In the above scheme, p decides the weights assigned to node itself and other neighbour nodes. When $p \to 0$, then information of node is not considered and for $p \to 1$, then information of neighbour is not considered.

2.3. ML-GCN with Semantic Values

In this approach we have used the model similar to the previous model with a change in generation of correlation matrix to improve the diversification ability of ML-GCN. The correlation matrix is generated with the use of semantic weights from Diverse Image Annotation (DIA) model [1].

2.3.1. Semantic Hierarchy, Synonyms and Weighted Semantic Path

Semantic Hierarchy (SH) and Synonyms

SH describes semantic dependencies among labels. The hierarchy is built upon 'is-a relationship' between labels. For example, we have two labels 'India' and 'country', then India 'is a' country so label 'country' will be ancestor of label 'India'. *Synonyms* are pairs of labels which denote same or similar meaning. For example, 'country' and 'nation' are *synonyms*. *SH* and *synonyms* are used for calculating semantic metrics and to ensure diversity during annotation in BIN-GCN. The *SH* and *synonyms* are obtained using WordNet[6].

Weighted Semantic Path (SP)

SP as described in [1] merges the idea of SH and Synonyms together. Various paths can be created among labels of our dataset, the paths differing only at synonyms are merged together as shown in figure 3. The semantic paths corresponding whole label set T is denoted by $SP_T = sp_1, sp_2, ...sp_m$. For each path, hierarchy layer and label weights are given. Layer value of leaf node is given 0 and the layer count increments till the root. Label weights are assigned to capture the information it represents in image. The weights must be assigned in such a way that (1) descendant label must have more weight then its ancestor label and (2) labels having more descendants must have less weight in comparison to labels having less descendants. The weight of a label y_i in path sp_j as shown in figure 3 is given as

$$w_{ij} = \tau^{l_{ij}} / |d_i| \tag{5}$$

where $|d_i|$ is number of descendants and l_{ij} is layer value of y_i in sp_j, and $\tau \in (0, 1)$ denoting decay factor between layers,we have $\tau = 0.7$. The weight of label y_i for whole set will be sum of its weights in all semantic paths. i.e., $w_i = \sum_j^{|SP|} w_{ij}$. For a subset Y of labels, the weights of label in SP changes, the leaf node in every path must have weight 1 (So change factor for that path is leaf weight) and all its ancestor weights are divided by the change factor as shown in figure 3.

2.3.2. Semantic Weight in ML-GCN

The correlation matrix created in ML-GCN doesn't take any consideration about diversity. **To introduce diversity we implemented ML-GCN using semantic weights in correlation matrix.** Suppose semantic weight of label i be $S[i]$ over the label set and P_{ij} be probability of j occurrence if i already have occurred as mentioned earlier. We update the conditional probability matrix as

$$\hat{P_{ij}} = P_{ij}(S[i]/S[j])^{\lambda} \tag{6}$$

where λ is a scalar. If $S[i] > S[j]$, then label i is more specific than label j (which is relatively more general). Therefore, label i should be preferred over label j, in the event of a clash between the two.

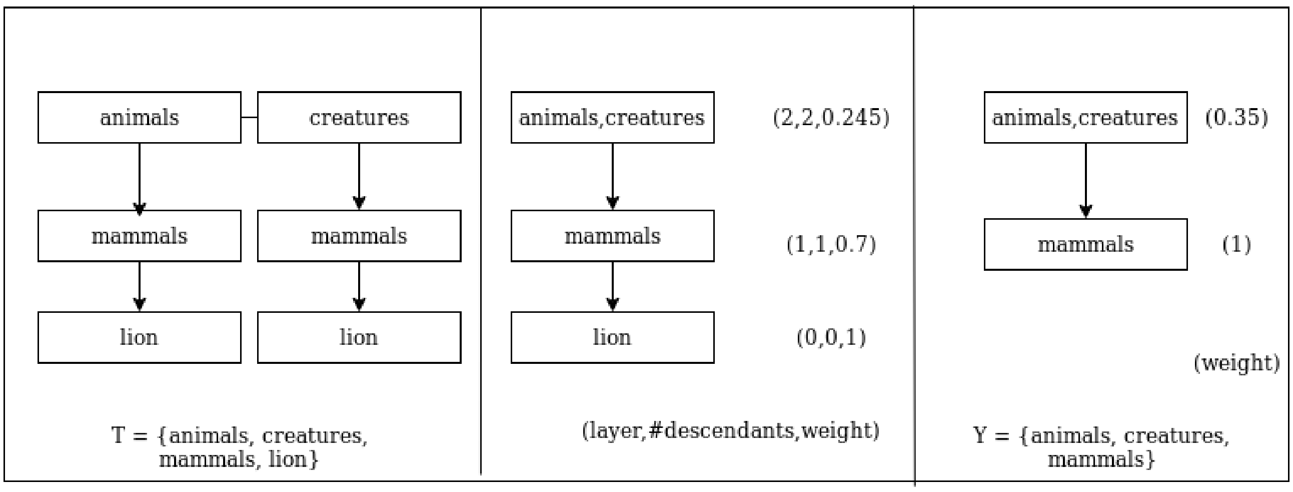

Figure 3. Left section shows merger of two semantic paths, Middle section shows how weights are calculated for whole set and Right section shows weights of particular subset

The weight propagating towards i from j in the label graph should therefore be increased, while the weight propagating towards j from i should be decreased. Hence, we multiply P_{ij} by $(S[i]/S[j])^{\lambda}$ which is greater than 1, and P_{ji} by $(S[j]/S[i])^{\lambda}$ (< 1). This transformation, in effect, increases the preference towards i in the resulting classifiers from the GCN.

2.4. Multilabel Classification using GCN-based Binary Classifiers (BIN-GCN)

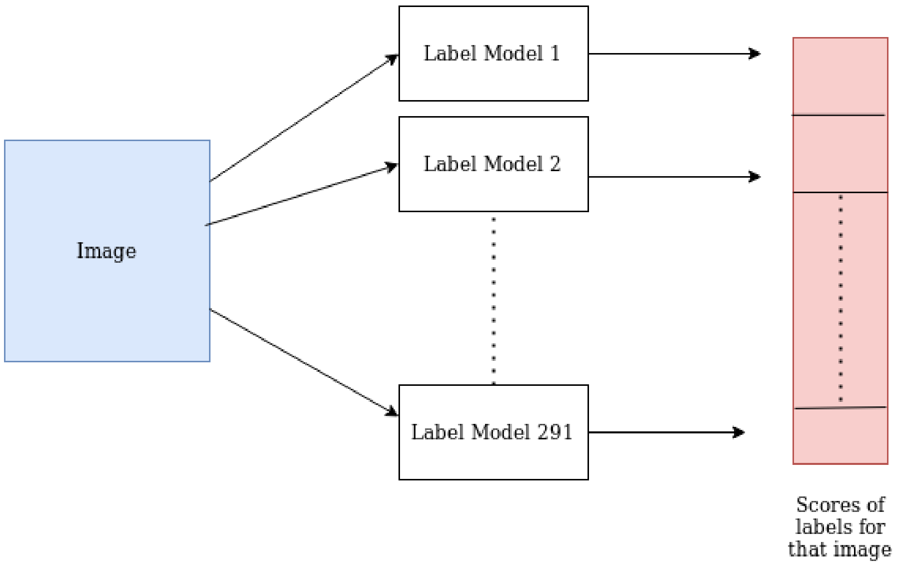

Figure 4. Flow of BIN-GCN with Binary Recognizior Model

In this approach, we have implemented multi-label classification using binary cassification model. The binary classification model uses GCN to aggregate the features of the nearest neighbours of an image, and transforms the aggregated features into a scalar score. In order to adopt the model for multilabel classification, we train separate models for each label, thus giving a total of 291 models for the IAPR TC-12 dataset. For each given test image, we run all the 291 models on that image, yielding a total of 291 scores, one corresponding to each label. We then employ and compare different sampling schemes to choose the final predicted subset of labels for the given image. The forward pass of one model (for one of the labels) is shown in Figure 5. We use the 536-dimensional VGGF-PCA features from [1] for the IAPR TC-12 dataset.

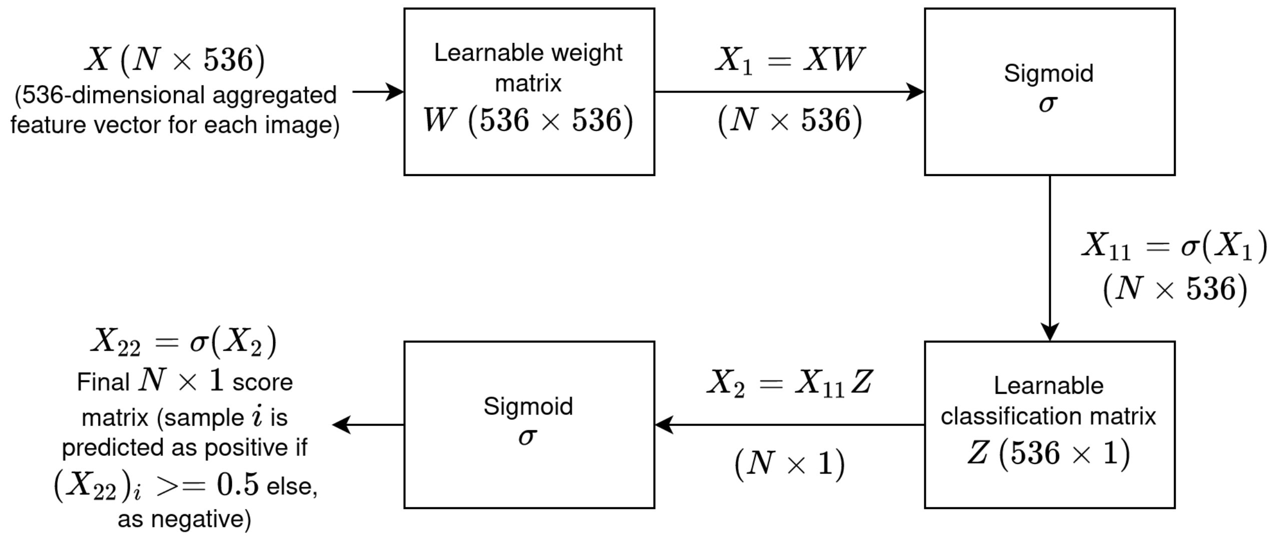

Figure 5. Forward pass of a single model (for one of the labels)

2.4.1. Generating the training set for each label

One of the major problems encountered during the process mentioned above is how do we generate the training data for each label from the multilabel data provided to us. Since we are training a separate model for each label, we need both positive and negative examples for a given label for feeding to its corresponding model. To this end, we use the *IterativeStratification* algorithm (Algorithm 1) from [8]. This algorithm takes as input the multilabel dataset D, the number of subsets k in which we need to divide the dataset, and a vector $r_1, r_2, ..., r_k$ of the proportions of the samples to be put into each subset. It outputs a set of disjoint subsets $S_1, ..., S_k$ of D. Each subset S_i contains approximately $r_i|D|$ samples, such that for any given label l, the number of samples with label l in subset S_i is approximately equal to r_i times the number of samples with label l in the whole dataset D. Hence, for all labels, the proportion of the samples containing a label is approximately equal in each subset, which ensures that no subset favours or ignores a particular label. This algorithm was originally proposed for $k-$fold cross validation on multilabel data, but we adopt it for sampling the training data for our models.

Our approach works as follows. For a given label l, we have to find a set of positive as well as a set of negative examples, with approximately equal sizes (so that the model doesn't get biased towards one of the outcomes). We first find all the samples in D containing the label l, which are basically the positive examples for label l. A parameter M dictates the maximum number of samples to choose (in our implementation, it is set to 2000). If the number of positive samples, $P_{l,total}$, is less than M, we choose all of them, else we randomly sample M examples from the positive samples. Thus, the sampled set of positive examples contains $P_l = min(M, P_{l,total})$ examples.

For sampling the negative examples, we could have used the same approach of randomly sampling M examples, but this causes a problem that some labels may not be considered, causing the model to not be able to learn to predict negatively in such cases. For example, if we are training a model for the label *phone*, but we do not include sufficient examples containing the label *laptop* (but not *phone*) in the negative examples, the model may never learn to not predict *laptop* for an example containing *phone*. Thus, we adopt the heuristic that sufficient number of samples for each label other than l should be included in the set of negative examples. To implement this, we use the semantic

Algorithm 1 IterativeStratification($D, k, r_1...r_k$)

```
    Input: A set of instances, $D$, annotated with a set of labels $L = \lambda_1, ..., \lambda_q$, desired number of subsets $k$, desired proportion of examples in each subset, $r_1, ..., r_k$ (e.g. in 10-fold CV $k = 10, r_j = 0.1, j = 1...10$)
    Output: Disjoint subsets $S_1, ..., S_k$ of $D$
    // Calculate the desired number of examples at each subset
 1: for $j \leftarrow 1, k$ do
 2:     $c_j \leftarrow |D|r_j$
 3: end for
    // Calculate the desired number of examples of each label at each subset
 4: for $i \leftarrow 1, |L|$ do
        // Find the examples of each label in the initial set
 5:     $D^i \leftarrow \{(x, Y) \in D : \lambda_i \in Y\}$
 6:     for $j \leftarrow 1, k$ do
 7:         $c_j^i \leftarrow |D^i|r_j$
 8:     end for
 9: end for
10: while $|D| > 0$ do
        // Find the label with the fewest (but at least one) remaining examples,
        // breaking ties randomly
11:     $D^i \leftarrow \{(x, Y) \in D : \lambda_i \in Y\}$
12:     $l \leftarrow \arg\min_i(|D^i|) \cap \{i : D^i \neq \Phi\}$
13:     for each $(x, Y) \in D^l$ do
            // Find the subset(s) with the largest number of desired examples for this
            // label, breaking ties by considering the largest number of desired examples
            // breaking further ties randomly
14:         $M \leftarrow \arg\max_{j=1...k}(c_j^l)$
15:         if $|M| = 1$ then
16:             $m \in M$
17:         else
18:             $M' \leftarrow \arg\max_{j \in M}(c_j)$
19:             if $|M'| = 1$ then
20:                 $m \in M'$
21:             else
22:                 $m \leftarrow randomElementOf(M')$
23:             end if
24:         end if
25:         $S_m \leftarrow S_m \cup \{(x, Y)\}$
26:         $D \leftarrow D \{(x, Y)\}$
            // Update desired number of examples
27:         for each $\lambda_i \in Y$ do
28:             $c_m^i \leftarrow c_m^i - 1$
29:         end for
30:         $c_m \leftarrow c_m - 1$
31:     end for
32: end while
33: return $S_1, ..., S_k$
```

paths from [1] and assume that a sample containing some label l also contains all the ancestors of l in the semantic hierarchy of the label-set (as proposed in [1]). Therefore, we identify some labels called negative labels, such that choosing samples containing these labels approximately covers all the labels (except l, for which we are building the model). For this, we choose the leaf nodes of all semantic paths (not containing l) and all the singleton nodes (not equal to l) as the negative labels. After we get this set of negative labels, which we denote by L_n, we find all the samples which have at least one negative label, but don't have the label l. Let's call this set of negative samples S_n. Now we have to sample from S_n, requiring that the sampled set should have the same size as the sampled positive example set, which is P_l $(= min(M, P_{l,total}))$ and the heuristic described above should be satisfied. For this, we use the *IterativeStratification* algorithm (Algorithm 1) with $D = S_n$, $k = 2$, $r_1 = P_l/|S_n|, r_2 = 1 - r_1$ and label-set $L = L_n$. The result would we two subsets of examples, $S_{n,1}$ and $S_{n,2}$, with the first subset approximately satisfying the two conditions stated above. Thus, we use $S_{n,1}$ as our final negative example set for the model corresponding to label l and discard $S_{n,2}$.

2.4.2. Aggregation of image features using GCN

We construct the graph of images $G = (V, E)$, where each image is represented by a node, as follows. For each node i, we find the $k-$nearest neighbour nodes based on the L_2 distances between the VGGF-PCA features for the images. Here, we choose $k = 5$. These $5-$nearest neighbour nodes of a node i become the adjacent nodes of i in graph G. Thus, each node in G has a degree of 5. Let the adjacency matrix of G be $\mathbf{A}$. We use the spectral propagation rule $\mathbf{X} = f(\mathbf{F}, \mathbf{A}) = \sigma(\mathbf{D}^{-1/2}\hat{\mathbf{A}}\mathbf{D}^{-1/2}\mathbf{FI})$ from [4], where $\mathbf{F}$ is the original $N \times 536$ feature matrix, $\mathbf{X}$ is the resulting $N \times 536$ aggregated feature matrix, $\mathbf{D}$ is the inverse square root diagonal degree matrix ($D_{ii} = (degree\ of\ node\ i)^{-1/2}$ and $D_{ij} = 0$ for all $i \neq j$), $\hat{\mathbf{A}} = \mathbf{A} + \mathbf{I}$, and we choose the transformation matrix to be the 536×536 identity matrix, $\mathbf{I}$. We can see that this single layer GCN is equivalent to the following transformation:

$$X_{ij} = \sum_{k \in N(i) \cup i} \frac{F_{kj}}{\sqrt{D(i)D(k)}} \tag{7}$$

where $N(i)$ is the set of the $5-$nearest neighbours of node i, and $D(i)$ is the degree of node i. Since all nodes have degree 5, the aggregated feature of the i^{th} node, $\mathbf{X}_i$ can be simply written as:

$$\mathbf{X}_i = \frac{1}{5} \sum_{k \in N(i) \cup i} \mathbf{F}_k \tag{8}$$

While training, aggregated features $\mathbf{X}$ are generated in this way by choosing nearest neighbours from the training set, and then $\mathbf{X}$ is passed as the input to the forward pass procedure illustrated in Figure 5, which outputs an array $\mathbf{X}_{22}$ of predicted scores for each image. The mean squared error loss L is then computed using eq. (3) and backpropagated using gradient descent to train the weights of the model ($\mathbf{W}$ and $\mathbf{Z}$). We report the results for 2500 epochs. (Here $\mathbf{Y}$ is the ground-truth boolean array of whether the samples in the training set are positive or negative with respect to the label corresponding to this model.)

$$L = \frac{1}{2N} \sum_{i=1}^{N} ((X_{22})_i - Y_i)^2 \tag{9}$$

2.4.3. Choosing labels from predicted scores

After the training is complete, we run all the 291 models for a given test image. After the forward pass (depicted in Figure 5 for one label) for all the labels is completed, we obtain 291 scores, one per label. Now we have to choose k labels (here, we choose $k = 5$ based on these scores. **We implement and compare two approaches to do so**. The first straightforward approach is to choose the top 5 labels with the highest scores. The second approach which is similar to the approach proposed in algorithm-1 of [1] involves choosing some highest score labels such that no two labels belong to the same semantic path. When there is a conflict between labels from the same semantic path, we choose the one which appears lowest in the semantic path (if there are synonyms, we randomly choose one from them). We expect this approach to help improve the diversity and reduce redundancy between the predicted labels, since we are not choosing multiple labels from the same semantic path, and choose only the most specific one from such labels (which appears lowest in the path). The lower labels in the path are specific versions of their (more general) ancestors in the path, thus choosing the lower labels covers their ancestors as well. The approach which we implemented can be explained as, initially we make annotation matrix binary, using the middle value of range of label scores for each image as threshold score. If there are more then 5 labels in annotation for that image, we randomly remove synonyms labels and then finally we check for any semantic parent-child relationship and if we find such relationship then the parent label is removed.

3. Evaluation Metrics

We use a combination of traditional and semantic metrics (from [1]) to compare the exhaustiveness and diversification powers of the four models described above.

Let the dataset be denoted by $D = \{(x_i, Y_i) : i = 1...N\}$ where x_i is the i^{th} image and Y_i is the set of labels assigned to the i^{th} image in the ground truth. Let the Z_i be the predicted set of labels for the i^{th} image by some model. Also, let W_i be the set of images having label i in the ground-truth, i.e, $W_i = \{x_j : i \in Y_j\}$, and V_i be the set of images predicted to have the label i by some model.

3.1. Traditional Metrics

The traditional metrics used for multilabel image classification are shown in Table 2. We have used both example-based and class-based metrics. The example-based metrics are computed for each example and their average is reported over all the test images. Similarly, The class-based metrics are computed for each label and their average is reported over all the labels.

3.2. Semantic Metrics

The traditional example-based metrics treat all the labels equally and independently, and focus on the correctness and exhaustiveness of the predicted labels. They do not take diversity into account. To overcome this problem, we use the semantic metrics proposed in [1], which evaluate representation and diversity jointly. These metrics are computed on the basis of semantic paths. The Algorithm 2 from [1] shows the procedure to compute these metrics for one image. We report the results averaged over all test images.

4. Datasets

Table 3 summarize the details of all the datasets used for our project. The models and their variations are compared with each other with respect to the IAPR TC-12 dataset. The image collection of the IAPR TC-12 Benchmark consists of 20,000 still natural images taken from locations around the world and comprising an assorted cross-section of still natural images. This includes pictures of different

Metric Name	Description	Formula
Average Example-based Precision (EP)	The average ratio of the number of correctly predicted labels to the total predicted labels.	$EP = \frac{1}{N}\sum_{i=1}^{N}\frac{\lvert Y_i \cap Z_i\rvert}{\lvert Z_i\rvert}$
Average Example-based Recall (ER)	The average ratio of the number of correctly predicted labels to the total labels in the ground-truth.	$ER = \frac{1}{N}\sum_{i=1}^{N}\frac{\lvert Y_i \cap Z_i\rvert}{\lvert Y_i\rvert}$
Average Example-based F_1 Score (EF1)	The harmonic mean of OP and OR. Conveys the overall example-based test accuracy.	$EF1 = \frac{2 \times EP \times ER}{EP + ER}$
Average Class-based Precision (CP)	The average ratio of the number of images correctly assigned a given label to the total predicted images with that label.	$CP = \frac{1}{N}\sum_{i=1}^{N}\frac{\lvert W_i \cap V_i\rvert}{\lvert V_i\rvert}$
Average Class-based Recall (CR)	The average ratio of the number of images correctly assigned a given label to the total images with that label in ground-truth.	$CR = \frac{1}{N}\sum_{i=1}^{N}\frac{\lvert W_i \cap V_i\rvert}{\lvert W_i\rvert}$
Average Class-based F_1 Score (CF1)	The harmonic mean of CP and CR. Conveys the overall class-based test accuracy.	$CF1 = \frac{2 \times CP \times CR}{CP + CR}$

Table 2. Traditional metrics used for evaluating multilabel classification models. The first 3 are example-based metrics, and the last three are class-based metrics.

Algorithm 2 Semantic Metrics (for one image)[1]

```
    Input: The ground-truth label subset $Y$ and the predicted label subset $Z$
    Output: Semantic precision, $P_{sp}$, semantic recall, $R_{sp}$, and semantic $F_1$ score, $F_{1,sp}$
 1: Construct the semantic paths $SP_Y$ and $SP_Z$ (as described in [])
 2: for each $sp_j \in SP_Y$ do
 3:     for each $y_i \in sp_j$ do
 4:         if $y_i \in Z$ then
 5:             $s_{y_i,j} \leftarrow \omega_{y_i,j}$
 6:         else
 7:             $s_{y_i,j} \leftarrow 0$
 8:         end if
 9:     end for
10:     $s_j \leftarrow \max_{y_i \in sp_j} s_{y_i,j}$
11: end for
12: $P_{sp} \leftarrow \sum_j^{|SP_Y|} s_j / |SP_Z|$
13: $R_{sp} \leftarrow \sum_j^{|SP_Y|} s_j / |SP_Y|$
14: $F_{1,sp} \leftarrow 2(P_{sp}.R_{sp})/(P_{sp} + R_{sp})$
15: return $P_{sp}, R_{sp}, F_{1,sp}$
```

Dataset name	Dictionary size	Total images	Train images	Test images
IAPR TC-12	291	19627	17665	1962
IAPR TC-12 with some images removed (from DIA[1])	291	19452	17495	1957
Corel5k	260	4999	4500	499
PASCAL VOC 2007	20	9963	5011	4952

Table 3. Number of total, train and test images, and dictionary size in all datasets used

sports and actions, photographs of people, animals, cities, landscapes and many other aspects of contemporary life[2].Unlike other similar databases, images in IAPR TC-12 are accompanied by free-flowing text captions and on an average has 4.7 keywords per image[5].

5. Results

We have compared all the four models described above with respect to the IAPR TC-12 dataset, using nine metrics, visually, the average example-based precision (EP), recall (ER) and F_1 score (EF1), the average class-based precision (CP), recall (CR) and F_1 score (CF1), and the semantic precision (SP), recall (SR) and F_1 score (SF1) (all these metrics are explained in section 3). For each model, we compare some variations of the model using the aforementioned nine metrics. For the baseline (KNN) model, we vary the way in which the two factors (mentioned in step 2 of the label-transfer algorithm presented in section 2.1.2) are combined. For simplicity, we denote the first factor (co-occurrence) as C and the second factor (local frequency) as F. We compare four ways to combine them, visually, $F \times C$, $F + C$, $F + 0.5C$ and $F + 1.5C$. For the BIN-GCN model, we compare two variations based on how we choose the final set of labels from the predicted label scores. In the first approach, which we call 'Top-5', we simply choose the top 5 labels with the highest scores. In the second approach, which we call 'Semantic Sampling', we sample from the labels with score greater than 0.5, if there are more than 5 such labels (else, we choose all of them). We use the semantic paths from [1] to remove redundant labels and end up with 5 or less labels per image. These two approaches are described in section 2.4.3. Finally, for the ML-GCN and semantic ML-GCN models, we compare two variations, again based on how we choose the final labels. In the first approach (Top-5), we choose the top 5 labels with the highest scores, and in the second approach (Zero-based), we choose all the labels with positive scores and reject those with negative scores. We have used the IAPR TC-12 dataset with some images deleted (see row 2 of Table 3) for all models except for ML-GCN and semantic ML-GCN (for them we have used the full version). Table 6 summarizes the results for the IAPR TC-12 dataset.

In addition, we report the results of the baseline (KNN) model (all four variations) on the Corel5k dataset, excluding the semantic metrics. We also report the results of the ML-GCN model (both variations) on the PASCAL VOC 2007 dataset, again excluding the semantic metrics. For the VOC dataset, we use the Top-3 variation instead of Top-5 since this dataset contains only 20 labels. Table 4 and Table 5 summarizes the results for Corel5k and VOC 2007 datasets respectively.

6. Conclusion

We tried variations of label-transfer algorithm by changing the factors F and C for Corel5k and IAPR TC-12. In general the results of $F \times C$ were better then the remaining 3 variations for both datasets.

Combination Method	EP	ER	EF1	CP	CR	CF1
$F \times C$	36.75	**51.89**	**43.03**	**26.29**	**31.54**	**28.68**
$F + C$	36.90	50.88	42.78	25.76	31.11	28.18
$F + 1.5C$	**36.94**	50.94	42.82	25.77	31.12	28.20
$F + 0.5C$	36.88	50.82	42.74	25.72	31.03	28.12

Table 4. Results of baseline (KNN) model on the Corel5k dataset. Four different combination schemes are compared for the two label-transfer factors (co-occurrence, C and local frequency F).

Variation	EP	ER	EF1	CP	CR	CF1
Top-3	43.68	**83.41**	57.34	44.91	**90.09**	59.94
Zero-based	**78.04**	75.71	**76.86**	**78.24**	66.05	**71.63**

Table 5. Results of ML-GCN model on the PASCAL VOC 2007 dataset. Two variations in choosing the final labels from scores are compared, visually, Top-3 and Zero-based.

Model	Variation	EP	ER	EF1	CP	CR	CF1	SP	SR	SF1
Baseline	$F \times C$	43.33	41.33	42.31	32.23	**22.01**	**26.15**	44.18	41.84	40.73
	$F + C$	43.27	40.89	42.05	32.31	21.85	26.07	44.10	41.35	40.53
	$F + 1.5C$	43.26	40.88	42.04	32.31	21.85	26.07	44.09	41.34	40.52
	$F + 0.5C$	43.29	40.90	42.06	**32.35**	21.85	26.08	44.12	41.37	40.56
ML-GCN	Top-5	45.61	**43.24**	**44.40**	31.03	20.02	24.34	48.22	**44.17**	**43.60**
	Zero-based	**49.21**	37.82	42.77	31.40	20.61	24.89	**51.18**	38.65	40.04
Sem. ML-GCN	Top-5	20.68	17.05	18.69	6.24	6.754	6.491	20.61	16.18	17.25
	Zero-based	8.42	23.94	12.46	8.61	18.44	11.74	8.530	23.62	11.98
BIN-GCN	Top-5	4.51	4.39	4.45	22.24	10.91	14.64	5.52	5.31	5.06
	Semantic Sampling	4.41	4.34	4.37	21.47	10.69	14.28	5.37	5.27	4.98

Table 6. Results of all models and their variations compared on the IAPR TC-12 dataset

We reported the results of ML-GCN[10] with the variation of selecting Top-k labels and Zero-based annotation for IAPRTC-12 and PASCAL VOC 2007 datasets. The Top-k label selection in general gives better results.

We tried to implement diversity with ML-GCN[10] by changing the method of generation of correlation matrix for IAPR TC-12 dataset for both the variations. We thought that changing the correlation matrix using semantic weights as mentioned in equation (6) should give better results for semantic metrics but the results for neither of the metrics improved showing that we need to build our correlation metrics with another approach to get better results then ML-GCN [10].

We implemented multi-label image classification using a binary classification model by generating separate models for each label in IAPR TC-12 dataset. This involved generating an appropriate training set for each model, which should be small in size (to allow for reasonable execution time) and at the same time contain enough examples for each negative label. This was accomplished through *IterativeStratification*. The results of class based metrics for BIN-GCN are better than Semantic ML-GCN but results of other two metrics of Semantic ML-GCN are better then our BIN-GCN. We also implemented semantic sampling similar to algorithm 1 in [1] to improve diversity of model but there is negligible change in all metrics.

Observing the results of IAPR TC-12 for all models and their variations for all metrics, ML-GCN[10] with Top-k annotation gives better results for semantic and example-based metrics, while Baseline model[5] gives better results for class-based metrics.

7. A Note for the Reader

The project was done in collaboration with Rajat Sharma(B18CSE042) and Priyanshi Jain(B18CSE043) whose project serve as a companion to ours. Both the groups have explored the models as mentioned in Section 2 and reported their results.

However there are various differences between our projects. We have used IAPR TC-12 dataset for comparing all our models and there variations. While they have reported the results of ESPGAME dataset. Also, we have tried variations of label transfer algorithm in baseline approach on Corel5k and IAPR TC-12 datasets. We also report the results of Top-k label selection for ML-GCN and Semantic ML-GCN.Also, our project have implemented *Iterative Stratification* for selection of negative images for training our label-models. We have implemented semantic sampling for selection of labels in an attempt to introduce diversity in BIN-GCN annotation.

References

[1] B. Wu, F. Jia, W. Liu and B. Ghanem. “Diverse Image Annotation”. In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017, 6194–6202.

[2] *IAPR TC-12 Benchmark*. URL: `https://www.imageclef.org/photodata`.

[3] Tobias Skovgaard Jepsen. *How to do Deep Learning on Graphs with Graph Convolutional Networks, Part 1: A High-Level Introduction to Graph Convolutional Networks*. URL: `https://towardsdatascience.com/how-to-do-deep-learning-on-graphs-with-graph-convolutional-networks-7d2250723780`.

[4] Tobias Skovgaard Jepsen. *How to do Deep Learning on Graphs with Graph Convolutional Networks, Part 2: Semi-Supervised Learning with Spectral Graph Convolutions*. URL: `https://towardsdatascience.com/how-to-do-deep-learning-on-graphs-with-graph-convolutional-networks-62acf5b143d0`.

[5] Ameesh Makadia, Vladimir Pavlovic, and Sanjiv Kumar. “A New Baseline for Image Annotation”. In: *Computer Vision – ECCV 2008*. Ed. by David Forsyth, Philip Torr, and Andrew Zisserman. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 316–329. ISBN: 978-3-540-88690-7.

[6] George A. Miller. “WordNet: A Lexical Database for English”. In: *Commun. ACM* 38.11 (Nov. 1995), pp. 39–41. ISSN: 0001-0782. DOI: `10.1145/219717.219748`. URL: `https://doi.org/10.1145/219717.219748`.

[7] Jeffrey Pennington, Richard Socher, and Christopher Manning. “GloVe: Global Vectors for Word Representation”. In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Doha, Qatar: Association for Computational Linguistics, Oct. 2014, pp. 1532–1543. DOI: `10.3115/v1/D14-1162`. URL: `https://www.aclweb.org/anthology/D14-1162`.

[8] Konstantinos Sechidis, Grigorios Tsoumakas, and Ioannis Vlahavas. “On the Stratification of Multi-label Data”. In: *Machine Learning and Knowledge Discovery in Databases*. Ed. by Dimitrios Gunopulos et al. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 145–158. ISBN: 978-3-642-23808-6.

[9] Verma, Yashaswi and Jawahar, C. V. “Image Annotation by Propagating Labels from Semantic Neighbourhoods”. In: *International Journal of Computer Vision* 121 (Jan 2017), 126–148. ISSN: 1573-1405. DOI: `10.1007/s11263-016-0927-0`. URL: `https://doi.org/10.1007/s11263-016-0927-0`.

[10] Zhao-Min Chen, Xiu-Shen Wei, Peng Wang and Yanwen Guo. “Multi-Label Image Recognition with Graph Convolutional Networks”. In: *CoRR* abs/1904.03582 (2019). arXiv: `1904.03582`. URL: `http://arxiv.org/abs/1904.03582`.