T.R.

GEBZE TECHNICAL UNIVERSITY

FACULTY OF ENGINEERING

DEPARTMENT OF COMPUTER ENGINEERING

MULTILABEL CBIR

TÜRKER TERCAN

SUPERVISOR
PROF. ERCHAN APTOULA

GEBZE
2022

T.R.
GEBZE TECHNICAL UNIVERSITY
FACULTY OF ENGINEERING
COMPUTER ENGINEERING DEPARTMENT

MULTILABEL CBIR

TÜRKER TERCAN

SUPERVISOR
PROF. ERCHAN APTOULA

2022
GEBZE

This study has been accepted as an Undergraduate Graduation Project in the Department of Computer Engineering on 20/01/2022 by the following jury.

JURY

Member
(Supervisor) : Prof. Erchan APTOULA

Member : Prof. Erdoğan Sevilgen

ABSTRACT

One of the cornerstones for content-based image retrieval **CBIR** is to learn the similarities between remote sensing **RS** images. Single-label retrieval is a common practice in most systems, meaning that each image is labeled with the most relevant semantic information about its content. Each image in our situation might have a variety of labels (i.e., multiple classes). Mapping the semantic similarity of images into an embedding (metric) space has been fairly popular recently and its called deep metric learning. The triplet selection approach is a well-known way of learning metric space. This technique seeks to choose images that are similar (positive) to and distinct (negative) to a reference image referred to as an anchor. However, developing a loss function that will enable metric learning with triplet selection for multi-label images is a challenging task. We tried address this challenge in two stages. To begin, choose a set of anchors that are distinct from one another, and then choose a set of positive and negative images that are the most hardest to compare to the anchor image and calculate loss with selected set of triplets.

Keywords: Content based image retrieval, remote sensing, single-label, multi-label, metric learning.

ACKNOWLEDGEMENT

I'd like to express my gratitude to my supervisor, Prof. Dr. Erchan Aptoula, for contributing to this project and guiding me through our weekly sessions, as well as to my professor, Prof. Dr. Erdoğan Sevilgen, for listening to me and guiding me through each meeting.

Additionally, during my education, I would like to express my thanks and devotion to my family, who provided me with unwavering support and knowledge, as well as to all of my professors who contributed to my development.

January 2022 **Türker Tercan**

CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

Latest developments in remote sensing (RS) technology have resulted in content-based rather than text-based information retrieval approaches. Content-based image retrieval (CBIR) attempts to find similar RS images in a large archive that have similar semantic information with respect to a query image. Any CBIR system performance depends on capabilities learn to distinctive image representations to explain semantic content of RS images. The majority of CBIR systems use single-label (Figure: 1.1) retrieval, in which each image is labeled with the most significant semantic information about its content. In multi-label (Figure: 1.2) retrieval systems, each image may have consist of variety of labels (i.e, multiple classes). Metric learning has recently acquired popularity as a method for learning the semantics of complex RS images, with the goal of mapping the semantic similarity of images to an embedding (metric) space. The triplet loss function approach is a well-known way of learning metric space. This technique seeks to choose images that are similar (positive) to and distinct (negative) to a reference image referred to as an anchor.

I attempted to design a loss function in this project that would enable metric learning with a triplet loss function for multi-label images with two steps: i) choose a set of anchors that are distinct from one another and, ii) choose a set of positive and negative images that are most hardest to compare to the anchor image and calculate loss with selected set of triplets.

Figure 1.1: Example of single labeled images from UC Merced Dataset[1].

Figure 1.2: Example of multi labeled images from UC Merced Dataset[1].

1.1. Related Works

G. Sumbul, M. Charfuelan, B. Demir, and V. Markl [2] proposed a CNN architecture called the shallow-convolutional neural network (S-CNN), this architecture can be used for model comparisons. Gencer Sumbul et al. [3] offered a novel technique for triplet selection (Relevant, Hard and Diverse Positive-Negative Image Selection (RHDIS)), and can be used for comparison tests. The benchmark archive UC Merced Land Use (UCMerced) archive [4] is used. MLRSNet [5] published by Q. Xiaoman et al. Lai et al. selects random based triplets to train images [6]. Exhaustive triplet selection is proposed by C. Zhou et al. [7]. It learns feature space using all possible triplets but considerably increases the overall number of triplets.

2. IMPLEMENTATIONS

2.1. Deep Neural Network Architecture

A configurable dataset is made up of a series of batch images. The batch images' features are extracted using a specified DNN model. Then calculate the euclidean distance between each image feature. The labels of batch images were retrieved and label distances between each image were determined. The anchor, positive, and negative sets are selected using specified triplet selection technique that uses feature and label distances. With chosen triplets, the triplet loss is calculated, and the model is updated by back-propagation.

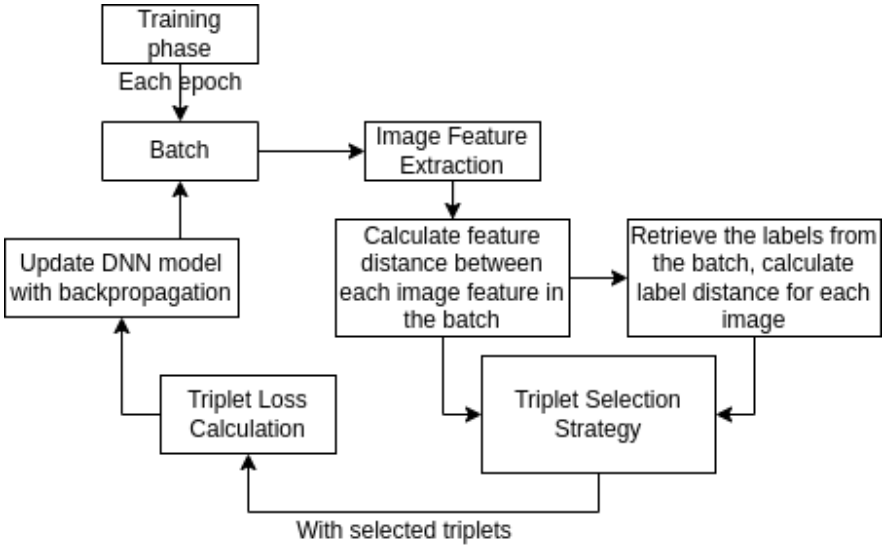

Figure 2.1: DNN Model architecture

2.2. Calculation Of Feature and Label Distances

According to the output feature size of the configured model and the number of images in the batch, the algorithm should construct the Euclidean distance matrix for the image's feature distance.

Algorithm 1 Calculate Pairwise Feature Distances

```
def calculate_pairwise_feature_distances(features):
    """
    Calculates pairwise feature distances with using the Euclidean
                                        distance matrix.
    :param features: Features of each batch image that are extracted
                                         from model (Shape(100,1024)
                                        number of batch images = 100,
                                        output feature size of the
                                        model = 1024)
    :return: pairwise feature distances (Shape(100, 100))
    """
    dot_product = tf.matmul(features, tf.transpose(features))
    square_norm = tf.linalg.diag_part(dot_product)
    distances = tf.expand_dims(square_norm, 0) - 2.0 * dot_product +
                                         tf.expand_dims(square_norm, 1
                                        )
    return tf.sqrt(distances)
```

To calculate label distances between each image, the Euclidean distance matrix should be calculated.

Algorithm 2 Calculate Pairwise Label Distances

```
def calculate_pairwise_label_distances(labels):
    """
    Calculates pairwise label distances with using the Euclidean
                                        distance matrix.
    :param labels: Two dimensional vector to represent each image's
                                        labels (Ex: Shape(100, 17)
                                        batch_size = 100, number of
                                        classes = 17)
    :return: pairwise label distances (Shape(100, 100))
    """
    dot_product = tf.matmul(labels, tf.transpose(labels))
    square_norm = tf.sqrt(tf.linalg.diag_part(dot_product))
    norms = tf.expand_dims(square_norm, 0) * tf.expand_dims(
                                        square_norm, 1)

    result = tf.minimum(1.0, tf.divide(dot_product, norms))
    return tf.subtract(1.0, result)
```

2.3. Triplet Selection Strategy

2.3.1. Triplet Loss

Triplet loss is a technique used by DML systems to acquire learn the metric spaces of triplet images [8]. The objective is to maximize the distance between the anchor and the negative image while minimizing the distance between the anchor and the positive image. It ensures that the positive image is closer to the anchor than the negative image at least by a margin.

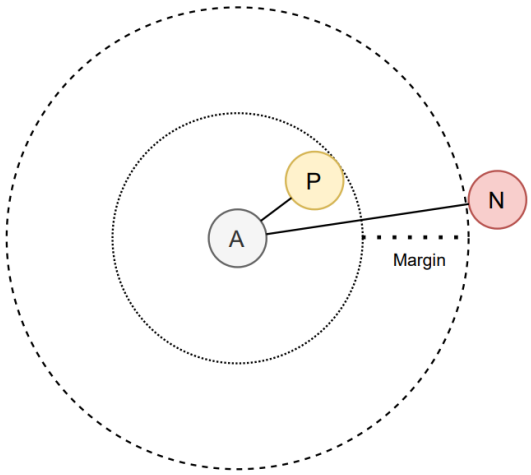

Figure 2.2: Abstract representation of triplet images. A= Anchor image, P= Positive image, N= Negative image

2.3.2. Calculate Triplet Loss

Algorithm 3 Calculate Triplet Loss

```
def calculate_loss(feature_distances, label_distances, margin):
    positive_dist = tf.expand_dims(feature_distances, 2)
    negative_dist = tf.expand_dims(feature_distances, 1)
    triplet_loss = positive_dist - negative_dist + margin
    mask = triplet_selection() # TripletSelection strategy that
                                    returns three-dimensional
                                    matrix
    triplet_loss = tf.multiply(tf.cast(mask, tf.float32),
                                    triplet_loss)
    triplet_loss = tf.maximum(triplet_loss, 0.0)
    return triplet_loss
```

2.3.3. Selection of Hard Triplets

To address the challenge of triplet selection, hard triplet selection might be the most efficient method of updating the model. If the distance between the anchor and the positive image is more than the margin and the positive picture is located farther away from the anchor image, this is referred to as a hard positive image. If the negative image is very close to the anchor image, it is referred to as a hard negative image. Figure 2.3 is a hard triplet example. These triplets will be insufficiently informative and will result in a high loss value when updating the model parameters.

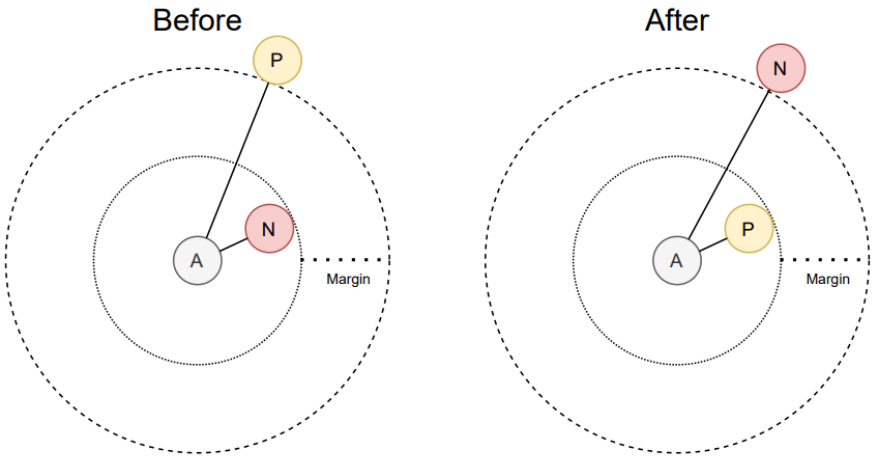

Figure 2.3: Abstract representation of hard triplet images. $\mathcal{A}$= Anchor image, $\mathcal{P}$= Positive image, $\mathcal{N}$= Negative image. The triplet drastically violates the margin and results in a major error.

The selection of triplet images which consist of multi-labels makes selecting more complex compare to the single-label selection. In our scenario, each training image labeled with at least one class label. Let $L = \{1, 2, ..., N\}$ be the set of all possible class labels and N is the size of multi-label classes that are available. Each image has been associated with multi-label vector $L_j = \{L_j^1, L_j^2, ..., L_j^N\}$. If the class label i is associated with image j $L_j^i = 1$, otherwise 0.

I tried to overcome this problem with 2 stages. i) Selection of small set anchors that are diverse from each other of batch images in feature space[3]. ii) Selection of hard positive and hard negative images in both feature and label space dimensions with respect to each anchor that results significant error.

Algorithm 4 Select diverse anchors in feature space

```
def select_diverse_anchors(distances, num_anchors):
    batch_size = len(distances)
    anchor_indices = np.zeros(num_anchors, dtype=np.int)
    first = np.random.randint(low=0, high=batch_size, size=1)
    anchor_indices[0] = first
    for i in range(1, num_anchors):
        local_max = np.zeros(batch_size)
        selected_anchors = anchor_indices[:i]
        for j in range(batch_size):
            if j in selected_anchors:
                continue
            local_max[j] = distances[j, selected_anchors].max()

        anchor_indices[i] = np.argmax(local_max)
    zeros = np.zeros((batch_size, batch_size, batch_size))
    zeros[anchor_indices, :, :] = 1
    return tf.cast(tf.constant(zeros), tf.bool)
```

Algorithm 5 Select hardest positive and negative images

```
def balanced_distance_selection(features, feature_distances, labels,
                                label_distances, loss_obj):
    norm_feat_dist = tf.divide(feature_distances, tf.reduce_mean(
                                feature_distances))
    norm_label_dist = tf.divide(label_distances, tf.reduce_mean(
                                label_distances))
    balanced_distances = tf.add(norm_feat_dist, tf.multiply(
                                norm_label_dist, alpha))
    batch_size = len(features)
    mask = select_all_valid_triplets(label_distances)
    mask = tf.cast(mask, tf.int32).numpy()
    for a in range(batch_size):
        sorted_distances = np.sort(balanced_distances[a, :])[-
                                    num_elements:]
        distances = sorted_distances[-num_elements:]
        for p in range(batch_size):
            if balanced_distances[a, p] not in distances:
                mask[a, p, :] = 0
        distances = sorted_distances[:num_elements]
        for n in range(batch_size):
            if balanced_distances[a, n] not in distances:
                mask[a, :, n] = 0
    return tf.cast(tf.constant(mask), tf.bool)
```

Step i: Select Diverse Anchors The method 4 of finding diverse anchors begins by randomly selecting an index from a batch of images. From feature distances, seeks out the most diversified anchors. Selects the maximum distance and continues until the specified number of anchors is satisfied.

Step ii: Select Hardest Triplets The method 5 for selecting the hardest positive and negative triplets is responsible for identifying the triplets that will result in a considerable loss of value. It begins by normalizing both feature and label distances on a mean basis. Then, the normalized label distance is multiplied by the supplied alpha value. Alpha is configured because if the number of class labels is big, label distances may be extremely small while calculating the hardest triplets. Distances between features are added in addition to label distances. After that, sort the totaled distances in relation to each anchor. Following sorting, the last elements of sorted distances will be hard positive images, whereas the starting elements will be hard negative images.

3. EXPERIMENTS

The first benchmark test is conducted using the UC Merced Land Use (UCMerced) archive. It has 2100 images, each of which is 256 by 256 pixels in size. Chaudhuri [1] annotated multi-labels. There are seventeen classes, and each image may have up to seven distinct classes assigned to it. Dataset seperated into 1260 train images, 420 validation images and 420 test images. The MLRSNet dataset [5] was used for the second benchmark test. It containns 109,161 remote sensing images that have been categorised into 46 categories. There 60 predefined class labels, and each image may have up to 13 classes assigned to it. The images have fixed size of 256 x 256 pixels. Randomly selected 25000 images from the MLRSNet dataset were divided into 10000 train images, 2500 validation images, and 12500 test images.

Margin parameter is set to 2.0. The number of anchors and the size of the triplets that are selected from the strategy are both set to 10. Two different CNN architectures are used. The shallow convolutions neural network (S-CNN) [2], and ResNet50V2 neural network architecture. Without pre-training, all models are assesed. Adam optimizer was used to train for 100 epochs. The 10 and 30 most retrieved similar images for both dataset are used to evaluate the metric calculations [1]. All test were done using the Google Colab service.

Random Selection The strategy for the select set of anchors, positive, negative triplets in the batch without any presumption.

All Possible Selection The approach is to choose all possible triplet combinations, including anchor, positive, and negative images. Finally, the chosen triplet size results in a large number.

Smart (RHDIS) Selection The strategy proposed by Gencer Sumbul et al. [2]. It select the triplets by evaluating the relevancy, hardness and diversity between anchors and triplets in batch of images.

Hardest Triplet Selection The triplets are chosen by considering the hardest label and feature distances in the batch of images. Alpha value is set to 5 in all experiments.

Table 3.1: The Performance of Triplet Selection Strategies For UC Merced Dataset with Two Different DL Model.

Method	**Model**	**Metrics(%)**			
		Precision	Recall	F1	Accuracy
Random Selection	ResNet50V2	67.676	71.551	69.555	56.113
	SCNN	63.402	53.764	58.178	45.093
All Possible Selection	ResNet50V2	68.731	72.909	70.754	56.985
	SCNN	58.105	44.91	50.655	38.074
Smart Selection	ResNet50V2	73.292	72.98	73.135	60.284
	SCNN	67.683	59.638	63.405	49.557
Hardest Triplet Selection	ResNet50V2	50.123	40.819	44.994	33.182
	SCNN	33.682	22.428	26.983	19.965

Table 3.2: The Performance of Triplet Selection Strategies For MLRSNet Dataset with ResNet50V2 DL Model.

Method	**Metrics**			
	Precision	Recall	F1	Accuracy
Random Selection				
All Possible Selection	53.53	55.511	54.503	41.814
Smart Selection	52.746	54.652	53.683	41.281
Hardest Triplet Selection				

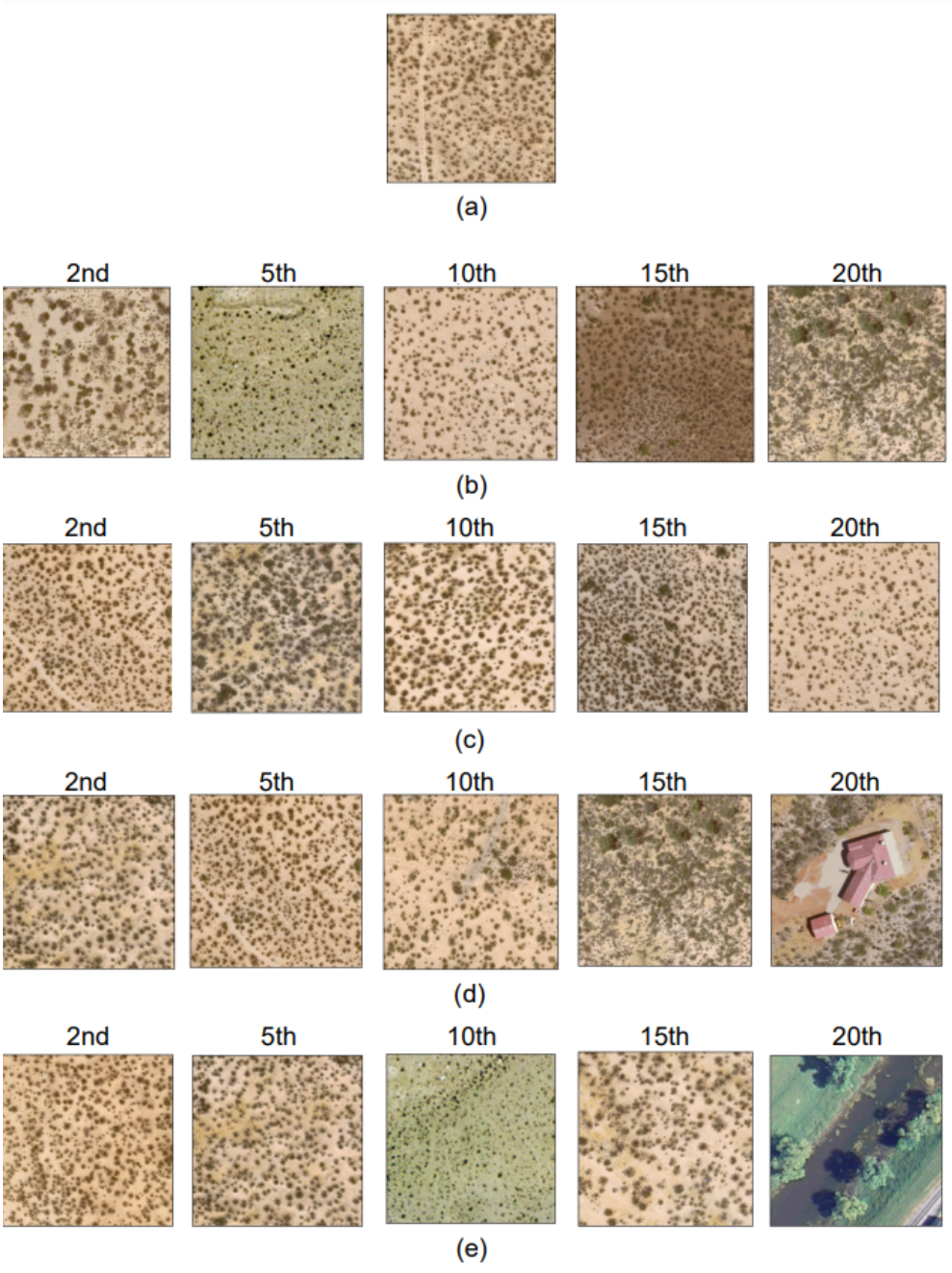

Figure 3.1: An image retrieval example using ResNet50V2 architecture: (a) query image; (b) images retrieved by Random Selection; (c) images retrieved by All Possible Selection; (d) images retrieved by Smart Selection [2]; (e) images retrieved by Hardest Triplet Selection (UC Merced Dataset)

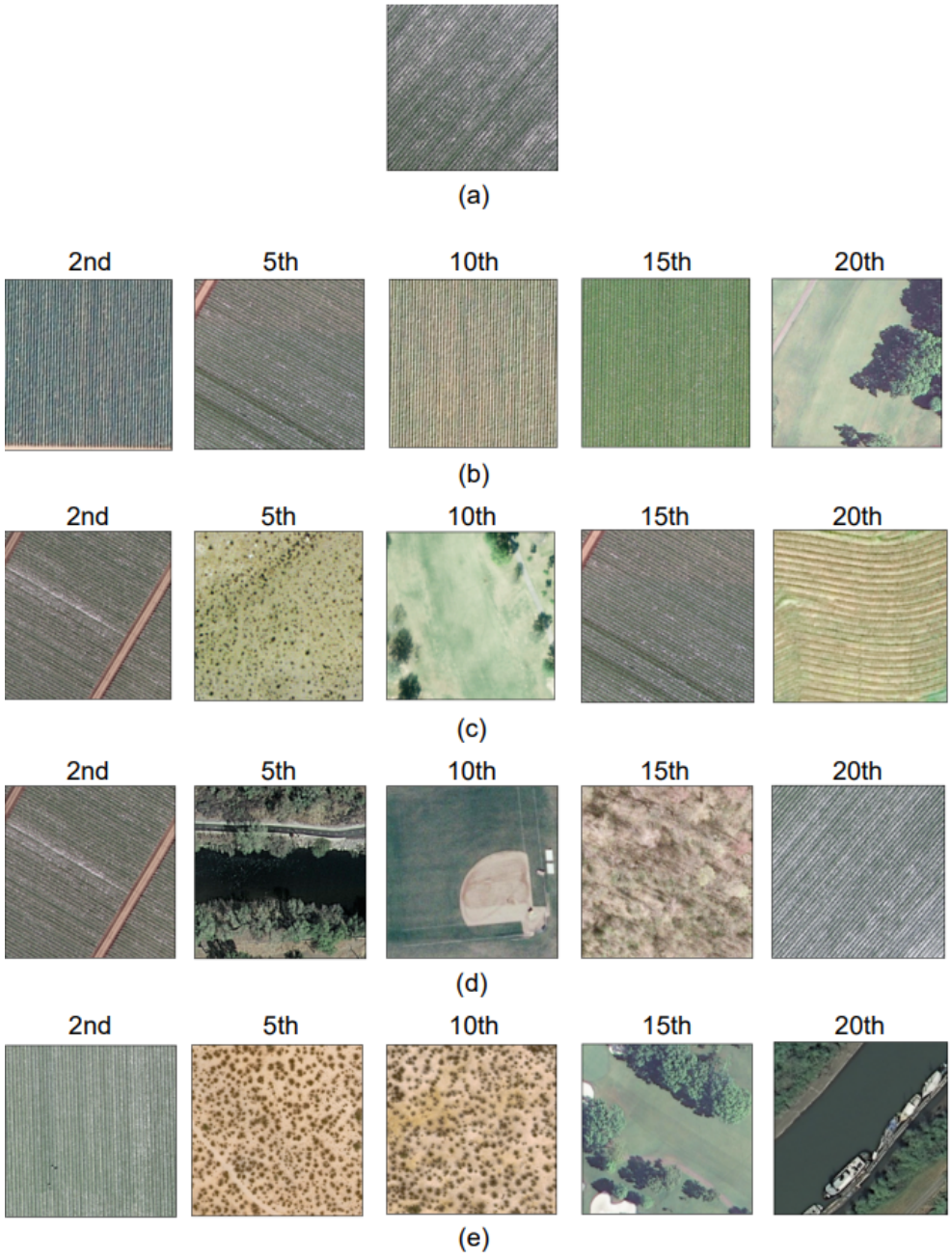

Figure 3.2: Another image retrieval example using ResNet50V2 architecture: (a) query image; (b) images retrieved by Random Selection; (c) images retrieved by All Possible Selection; (d) images retrieved by Smart Selection [2]; (e) images retrieved by Hardest Triplet Selection (UC Merced Dataset)

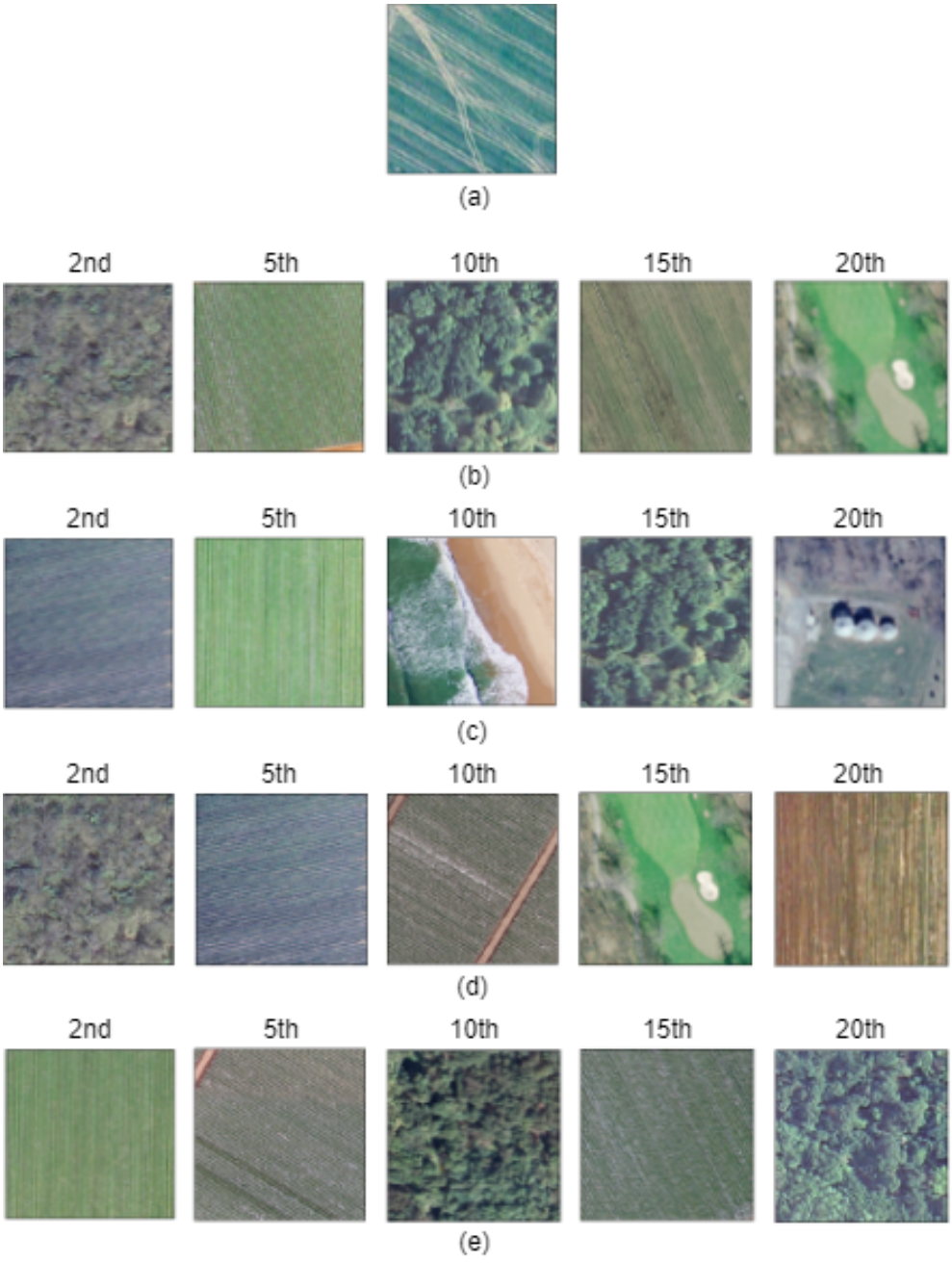

Figure 3.3: Another image retrieval example using S-CNN[1] architecture: (a) query image; (b) images retrieved by Random Selection; (c) images retrieved by All Possible Selection; (d) images retrieved by Smart Selection [2]; (e) images retrieved by Hardest Triplet Selection (UC Merced Dataset)

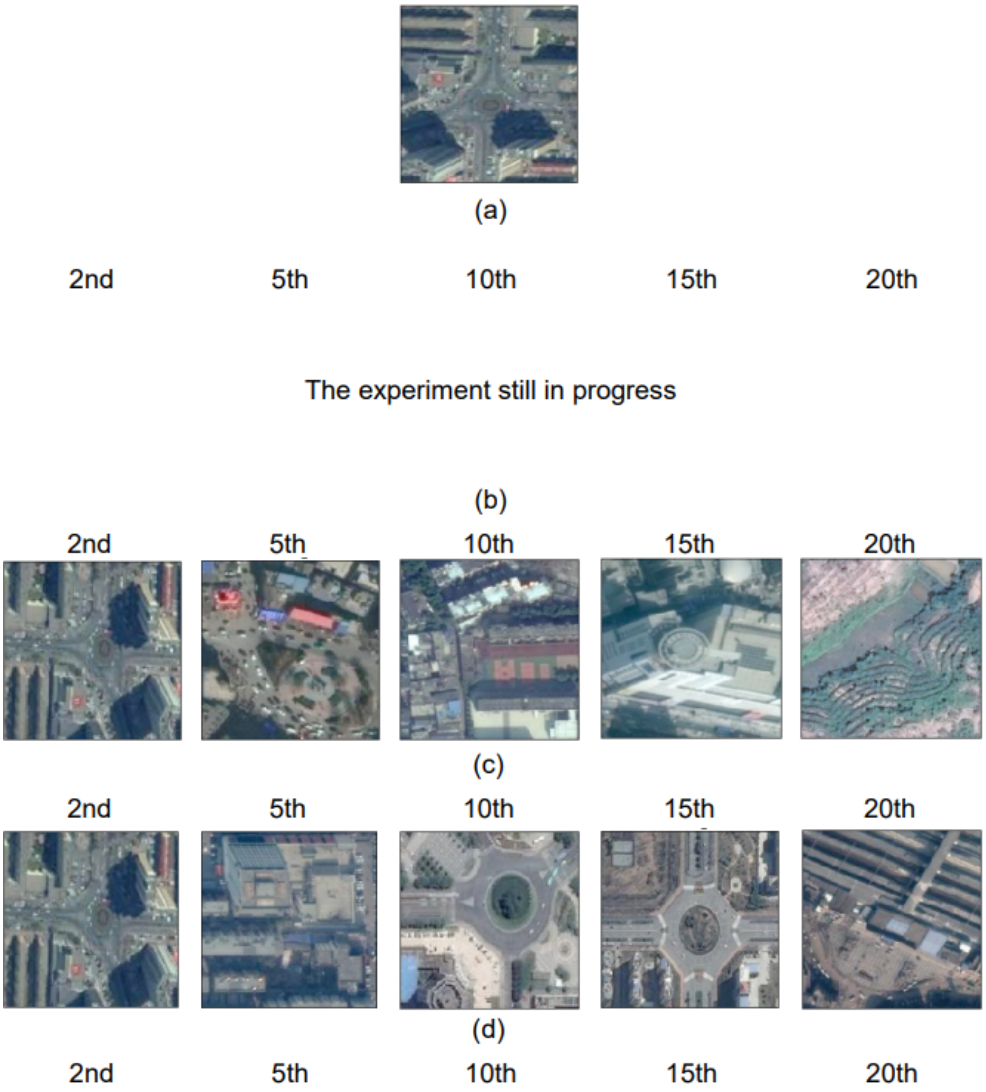

Figure 3.4: Another image retrieval example using ResNet50V2 architecture: (a) query image; (b) images retrieved by Random Selection; (c) images retrieved by All Possible Selection; (d) images retrieved by Smart Selection [2]; (e) images retrieved by Hardest Triplet Selection (MLRSNet Dataset)

The results of this experiment 3.1 demonstrated clearly that ResNet50 DL architecture is better than S-CNN architecture on all metrics. Both Random and All Possible selection strategies have shown very similar, but successful, outcomes. Even though their architectural design is rather straightforward in comparison to the other strategies. Smart selection [2] results clearly showed is the most suitable strategy for selecting the triplets. The hardest triplet selection is the worst strategy for UC Merced Dataset.

4. CONCLUSIONS

I attempted to develop a metric learning method with triplet loss and a triplet selection strategy that would operate on multi-labeled datasets for this project. The first phase is to select diverse anchor images, and the second step is to select the hardest triplets based on their feature and label distances. The experiment results indicate that all of the training strategies are highly dependent on the dataset's multiclass labels. A suitable technique for selecting triplets for multi-label datasets should be using both semantic content and image label information. Very good example of that is the smart selection which is proposed by G. Sumbul et al. [2].

BIBLIOGRAPHY

[1] B. Chaudhuri, B. Demir, S. Chaudhuri, and L. Bruzzone, "Multilabel remote sensing image retrieval using a semisupervised graph-theoretic method," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 56, pp. 1144–1158, 2018.

[2] G. Sumbul, M. Charfuelan, B. Demir, and V. Markl, "BigEarthNet: A large-scale benchmark archive for remote sensing image understanding," *IEEE International Geoscience and Remote Sensing Symposium*, pp. 5901–5904, 2019.

[3] G. Sumbul, M. Ravanbakhsh, and B. Demir, "Informative and Representative Triplet Selection for Multilabel Remote Sensing Image Retrieval," 9 November 2021. [Online]. Available: `https://arxiv.org/abs/2105.03647`.

[4] Y. Yang and S. Newsam, "Bag-of-visual-words and spatial extensions for land-use classification," *International Conference on Advances in Geographic Information Systems*, pp. 270–279, 2010.

[5] Q. Xiaoman, Z. Panpan, W. Yuebin, *et al.*, "MLRSNet: A Multi-label High Spatial Resolution Remote Sensing Dataset for Semantic Scene Understanding," 2021. [Online]. Available: `https://data.mendeley.com/datasets/7j9bv9vwsx/3`.

[6] S. Roy, E. Sangineto, B. Demir, and N. Sebe, "Metric-learning-based deep hashing network for content-based retrieval of remote sensing images," *IEEE Geoscience and Remote Sensing Letters*, vol. 18, pp. 226–230, 2021.

[7] C. Zhou, L. Po, W. Y. F. Yuen, *et al.*, "Angular deep supervised hashing for image retrieval," *IEEE Access*, vol. 7, pp. 127 521–127 532, 2019.

[8] F. Schroff, D. Kalenichenko, and J. Philbin, "Facenet: A unified embedding for face recognition and clustering," *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 815–823, 2015.