Using explainability to aid Active Learning

Ananya Aggarwal
Dhirubhai Ambani International School
ananya.aggarwal@gmail.com

Stephan Lemmer
University of Michigan, Ann Arbor
lemmersj@umich.edu

Abstract

Deep neural networks have improved the state-of-the-art machine learning methods but they are notoriously data hungry and require many instances of training data. Active learning works to select the optimal collection of training instances from all available training instances. However, current active learning methods do not take into account human perspective. This human perspective allowed us to split up NN understandings into two styles: misidentification of features and misclassification of accurately identified features. The latter is an error that we believe can be resolved by additional relevant data. Therefore the proposed algorithm first identifies instances of the latter error and then uses an active learning approach to reduce the error. We compared our proposed algorithm to a random selection baseline and found that the proposed method averaged an additional increase of 2% in accuracy of an image classification NN.

1. Introduction

One of the biggest hurdles with most machine learning models is the requirement of large labelled data sets [6, 13]. It is difficult to acquire a large data set because of the high annotation costs associated with it which make the development of the model impractical. A common solution to this problem is active learning [20, 5]. Active learning algorithms develop a method for the model to decide which training examples should be added to the data set and thus labelled while maintaining high accuracy. The training examples are selected based on what the model is yet to learn, or the gap in the model's knowledge. We refer to a gap in the model's knowledge as types of data that confuses the model - certain classes, angles, shapes, etc.

Pool-based active learning algorithms work in a bootstrapping manner wherein the algorithm starts with an initial training data set, a subset of the universal training data set[19]. Bootstrap sampling involves the extraction of sample data frequently with replacement from a data set to estimate a population parameter. Using a selector or acquisition function, the algorithm iterates through excluded training examples to select examples to be incorporated into the next iteration of the training data set. A selector function decides which data points are worth labeling, reducing the number of data points requiring a label. The model is then retrained on the generated training data set. This process is repeated p times. The final training data set is often smaller in size than the universal training data set, reducing the number of points to be annotated in turn reducing annotation costs. The selector or acquisition function plays a pivotal role in the final training set that is created [8]. There are two broad categories of selector functions: input distribution and model uncertainty. Both selector functions aim to identify what the model doesn't know and then close the gap in the model's knowledge. The former aims to diversify the training data set by finding dissimilar images in the universal data set via distance computations. Therefore, this method is easier to implement and faster. The latter method leverages the model's predictions to create a robust data set by adding the samples with the highest uncertainty. This makes the method more effective but more computational heavy [19, 14].

Instead of looking at improvements to the selector functions, this paper focuses on a novel way to identify which data points are optimal to use as baselines to select additional data points from the unused data set. In other words, this paper explores the identification of baseline data points to use as inputs for a selector function. These data points represent the gap in the model's knowledge of the model and thus when we add additional data points basis the gap, we address the it by providing information that will help close the gap. We will look at the task of image classification, with a maximum of 500 data point additions. Importantly, we leverage explanations from the model to identify the images in which the model has identified the correct features (areas of the image with the object) but misclassified the image. We then employ the concept of input distribution to identify the most similar images and add them to the data set. This sort of methodology fuses the input distribution and model uncertainty ideology and benefits from the convenience of input distribution and effectiveness of model

uncertainty.

2. Related Works

2.1. Active Learning

Active learning is broadly divided into two types: pool-based and stream-based [21]. The explanations above were for pool-based active learning. In stream-based active learning, each example is selected in a streaming manner wherein the decision on whether to label the example or not needs to be made instantly. Stream-based active learning is theoretically guaranteed to be more effective [9]. Often, stream-based learning is considered a subset of pool-based learning because a pool can be turned into a stream.

In pool-based active learning, we have a labelled data set, D_l, and an unlabelled data set, D_u. It is important to note that D_u is relatively larger than D_l. The universal training data set, D, is the union of D_u and D_l. Therefore, $D = x_1, x_2, x_3, ..., x_{n_u}, (x_{n_u+1}, y_{n_u+1}), ..., (x_n, y_n)$, where $x_i \in \mathbb{R}^d$ and $y \in \mathbb{Z}$. D_u contains the first n_u unlabelled instances and D_l contains the next $n - n_u$ labelled instances. Then a neural network is trained on data set D_l. For p iterations, the selector function considers D_l, D_u and current neural network and outputs x_j such that $x_j \in D_u$. The selected x_j is then annotated and then added to D_l [9]. This is repeated until the data set is completed and the final training data set is much smaller than D, reducing annotation costs.

These selector functions are of two types in pool-based active learning - model uncertainty or certainty sampling and input distribution [8]. Model uncertainty trains the model after every addition to D_l and runs the trained on D_u to identify the samples the neural network is most unsure of (least certain) and adds them to the data set [11]. In this framework, an active learner queries the instances about which it is least certain how to label. [12, 19]. Input distribution is less common because it doesn't consider the model's predictions and instead aims to minimize the similarity between data points of the data set. Input distribution is an easier implementation, in terms of time and computation, whereas model uncertainty is more effective solution [14].

Both these methods offer solutions to the proposed issue of a gap in the model's knowledge because it allows the model to learn what it doesn't know by adding the most relevant and effective data points to the data set.

However, the proposed method adds a human aspect that we believe most active learning algorithms lack. Our first and foremost step is to identify and understand the gaps in the model's knowledge using explainability. Using this understanding, we try to add the most relevant datapoints that will close this gap.

2.2. Explainable AI

AI is edging towards becoming an integral part of our society, be it our Spotify recommendations, the content our social media feeds present to us or the advertisements we are shown on our devices. However, as the field of computer vision develops, convolutional neural networks (CNNs) will play a pivotal role in more consequential decisions, like medical diagnosis and decisions of an autonomous cars [7]. Thus, it becomes important to demystify the 'black box' of CNNs and make our models more interpretable [1, 2]. Interpretability refers the transparency of a model or the ability of the developer to discern why the model predicts what it predicts, allowing the developer to understand what the CNN has learned and its gap in knowledge [16, 4]. This gap can also be represented through a subtle bias of the model. Kaufman et al.[10] cite a medical situation wherein the patient ID was heavily influential on the predictions indicative of bias in the model which could, to some extent, be resolved if given model explanations. Burns et al.[3] cite an instance where the CNN used contextual clues like location to make predictions. Even though the CNN had high accuracy, they cited that it was important the CNN was right for the right reasons and didn't contain any egregious errors, like predicting the word "man" in a snowboarding image.

One of the biggest challenges that arises with explanations in the field of computer vision is that they must be interpretable by humans but also accurate representations of the model. For image classification, for example, humans may be able to comprehend a binary vector representative of the presence of an adjacent patch of similar pixels even though the CNN interprets the image as a three-channel tensor. [15] Usually, in the field of computer vision, the explanations are typically identifications of features or patches of the test image that were critical for the decision of the CNN.

Most importantly, explanability gives an insight into the knowledge of the model allowing us to comprehend what the model understands and what it gets confused with. As a result, we assert that explainability forms a strong foundation for active learning because it allows us to demystify the 'black box' and figure out the optimal data instances to add to the training data set.

We now know explore two popular methods in explainability, LIME and grad-CAM, to outline the state-of-the-art approaches to explainability. They differ in the sense that LIME is a model-agnostic approach while grad-CAM is tailored towards computer vision tasks including VQA, image detection and classification [16, 15].

2.2.1 Local Interpretable Model-agnostic Explanations (LIME)

LIME [15] aims discern an interpretable model from a set of possible interpretable representations which is locally faith-

ful, i.e. it is reflective of the model's behaviour in the region of the instance being analysed. LIME also puts special focus on remaining model-agnostic, i.e. maintaining no assumptions about the original model.

When applying LIME to CNNs for image classification, the LIME model learn positive and negative weights for all the super pixels of the image and all the positively weighted super pixels represent the explanations behind the CNN's prediction.

2.2.2 Gradient-weighted Class Activation Mapping (Grad-CAM)

Grad-CAM is a a method to extract visual explanations from deep neural networks through gradient-based localisation. It uses class-specific gradient information to localize important regions [18, 17]. Grad-CAM, in comparison to LIME, is specifically designed to generate visual explanations from a CNN without architectural alterations. Grad-CAM leverages the fact that convolutions features preserve spatial information unlike fully-connected layers. Therefore, grad-CAM assumes that the last convolutional layers is the best indication for explanations because it provides a strong trade-off between complexity and accuracy. The gradients of the last convolutional layer are used to discern the importance of each neuron of the layer for the prediction. [16]

Since grad-CAM has been designed specifically for computer vision tasks, including image classification and focuses on geographical isolation in an image, this paper will be using grad-CAM as the method to obtain our model's explanations.

3. Method

The proposed algorithm aims to leverage insights from the XAI method Grad-CAM to conduct active learning. We cap the number of additions at $maxAdditions$ and start with an initial training data set of 6000. The method aims to minimize the number of the data instances requiring labelling while increasing accuracy of the model on a unseen test data set. The following are the variables employed in the paper:

$\alpha =$ Hyperparameter that adjusts the strictness in selecting query images (number of nonblack pixels the red region of the Grad-CAM heat map lies on)

y_q = the class of the query image

$maxAdditions$ = Defined by the problem statement as the maximum number of data points that can be added. We use 500.

In this paper we present a novel approach to improve classic active learning methods by leveraging grad-CAM explanations. The proposed algorithm (refer Figure 1) can be divided into two parts - identification of *query images*(refer Figure 2) and data set additions basis the identified query images (refer Figure 5). Query images are the images we identify as representations of the gaps in the model's knowledge. Query images will give us an insight into the types of images the model typically misclassifies, allowing us to make more informed decisions about what kind of datapoints are worth adding.

3.1. Identification of query images

The first piece of our proposed algorithm is to use grad-CAM to identify query images. We start by splitting our train data sets into training set (X_{train}) and additional training set (X_{add}) in the ratio 1:9, and our test data set into potential query images (X_{query}) and testing pool (X_{test}) in the ratio 1:1. In order to understand the initial model, we trained a CNN on X_{train} and tested it on X_{query}. During the testing process, we saved all incorrectly classified images as potential query images because we believe incorrectly classified images are the best representations of the gaps in the model's knowledge. We call the incorrectly classified images potential query images because the CNN makes two broad types of mistakes - misidentification of features (refer Figure 3a) and misclassification of accurately identified features(refer Figure 3b). In this paper, we try to minimize the latter error by adding additional relevant data points.

Logically, the first task is to separate the potential query images into the two error types, misidentified features and misclassified yet accurately identified features. The insight into the model required for this separation is provided by grad-CAM. Grad-CAM gives us a heat map representative of the features of the image that were given maximum weightage, like in Figure 4. Therefor, we have a good idea of what regions of the image the model focuses on before classifying it. We identify misidentification of features through the Grad-CAM generated heat map. If the heat map surrounds the background, we assume the model's focus was the background rather than the object. Similarly, we identify misclassification of accurately identified features when the heat map is cast mainly on the object. This indicates that the model's focus was on the object, as it should be, meaning the features are identified correctly and thus the mistakes was in the classification of those features. Our query images thus become the ones where the heat map is cast onto the object. We assume a data set of single channel images on a solid black background. Thus when we gray scale the data set, pixels that have a value lesser than 0.5 are more white than black thus likely to belong to the object. We use this knowledge to discern whether the heat map lies on the correct features (the object).

We first extract the red channel of the heat map to un-

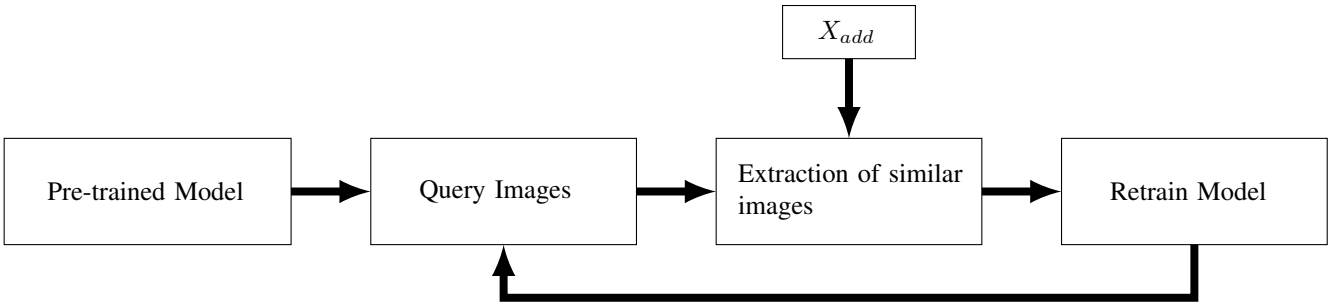


Figure 1: Higher level overview of the proposed algorithm.

derstand which pixels are the focus of the heat map. All the red pixels are then iterated through. For each red pixel we check if the corresponding pixel in the potential query images is gray. If this non black pixel count is $> \alpha$, we use the potential query images as a query image. As seen in figure 6, figures 6a, 6b and 6c have heat maps that are not on the object, or are of the misidentification of features error type. Figures 6d, 6e and 6f, on the other hand, have correctly identified features but are still misclassified by our model. Therefor, the counting of the non-black pixels works accurately in classifying the two error types. The latter error can be resolved using active learning processes, selecting more images to add.

3.2. Data set additions basis the identified query images

Now that we have a reference for what our model doesn't know (the query images), we can use them as a baseline to start adding data points. As explained previously, there are two main active learning heuristics: model uncertainty and input distribution. We use the ideology of the input distribution method in our paper because it allows us to take maximum advantage of our query images. Refer to Figure 5. Essentially, we add the images most similar to the query images.

We use cosine similarity to discern which images from X_{add} would be the most valuable additions to X_{train}. Let us assume y_q represents the class of the query image. Cosine similarity is calculated by

$$\cos(\theta) = \frac{A \cdot B}{|| A || || B ||}$$

where A and B represent the 2D vectors of the two images at hand. The higher the $\cos(\theta)$ value, the more similar the images. In order to select the data points to add, we find the $\frac{maxAdditions}{len(queryimages)}$ most similar image from X_{add} which are of class y_q. We move these images from y_{query} to y_{train} and retrain the model.

4. Experiment

4.1. Experimental Set-Up

In order to test the proposed algorithm, we use the FMNIST dataset. We chose this data set because the CNN trains relatively quickly on it, allowing us to conveniently retrain our model after every query image [22]. Additionally, since the images are already gray scaled, all background are of the same colour and we have a range of values for the pixels corresponding to the object allowing us to differentiate between the two aforementioned error types with ease.

For the CNN, we used LeNet-5 architecture because it is a well established CNN architecture that is designed for the MNIST dataset [23]. Our metric is accuracy and our baseline is random addition. For the random addition, we simply add the randomly selected images from X_{add} in the same manner as we did in the algorithm (retrain the model after addition of $\frac{maxAdditions}{len(queryimages)}$ images.

We ran the algorithm 3 times and then averaged the results. This was especially useful in the case of the random baseline because we noticed a high standard deviation.

5. Results

In this section we go over the results we obtained after running our experiment. We evaluate the success of our proposed algorithm bases the effect on the accuracy of the model when tested on the unseen test data set X_{test}.

Table 1 summarises the final values for accuracy after all

	Final accuracy with the Proposed Algorithm	**Final accuracy with Random Selection**
Trial 1	0.841	0.843
Trial 2	0.835	0.846
Trial 3	0.847	0.791
Average	**0.841**	**0.827**

Table 1: Final accuracies with the proposed algorithm and the baseline, random selection.

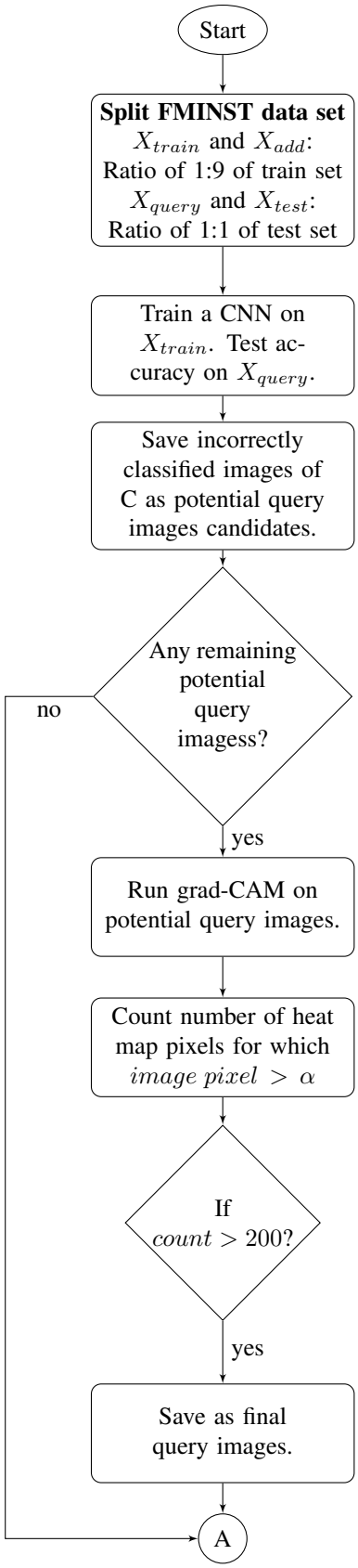
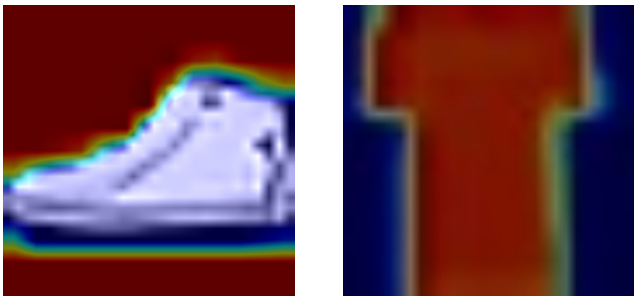


Figure 2: A flowchart outlining the proposed algorithm: identification of query images



(a) Misidentification of features.

(b) Misclassification of accurately identified features

Figure 3: Two broad types of errors made by a CNN.

the data instances have been added. As can be seen, our model does beat the baseline when averaged by approximately 2%. The final accuracy of the random baseline does fluctuate owing to the random nature of the algorithm. Our method does provide more consistent results. This is evident through the differences in the standard deviation of the final accuracy. The proposed algorithm has a deviation value of 0.0060 whereas the random baseline has a deviation value of 0.0309. This suggests that the proposed algorithm is a smarter method as each addition builds off the previous query image. The frequent fluctuations in random selection implies that is, in fact, random and a play on chance, therefore not very reliable.

Taking a look at Figure 7 gives us a more detailed insight in the accuracy values per addition iteration. As can be seen, at the beginning there is high fluctuations and the random algorithm does better briefly. However, after around 50 additions, the algorithm accuracy does consistently better than the random accuracy. The shaded region represents the deviation in the graphs and the regions do overlap, suggesting that the results have high uncertainty.

6. Discussion

6.1. Possible Improvements to the algorithm

The results may be improved because test X_{test} is of the same size as test X_{query}, in our experiment, which means there is a high likelihood of set D containing many novel

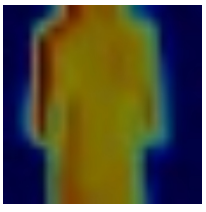


Figure 4: Example of grad-CAM output.

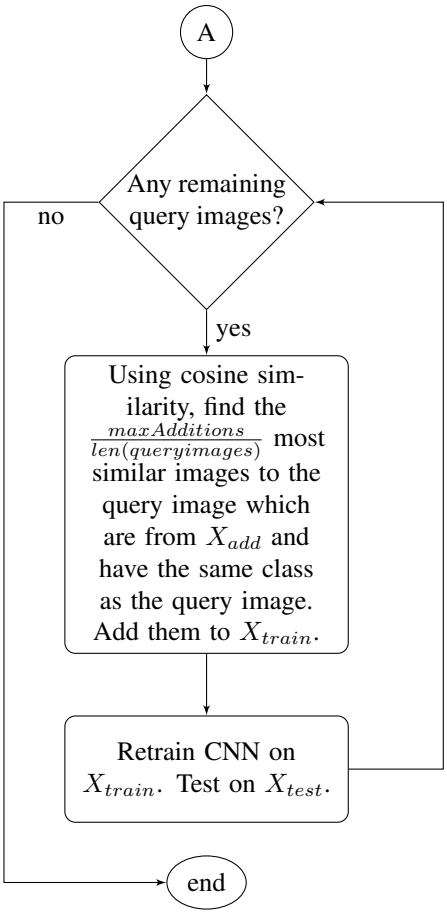


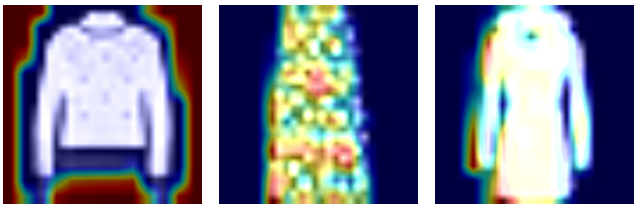
Figure 5: A flowchart outlining the proposed algorithm: Data set Additions basis the query images

error sources not present in X_{query} and vice-verse. We hypothesize adjusting the split ratio by increasing it towards the training side will help reduce the error because it will equip us with more data to understand the model.

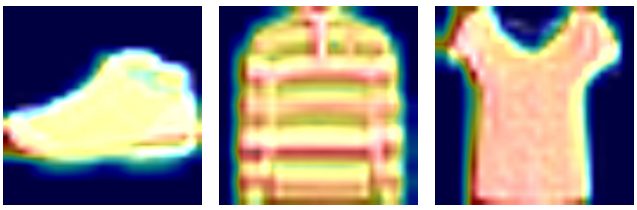
There are many ways we could have decided the number of images per query image. We could have customized the number of additions per query image, depending on how the severity of the mistake. The severity of the image could be gauged by the ratio of 50 most similar images in X_{train} that are of class y_q. A more advanced approach would include consideration of the rank of 50 images (in terms of similarity). This means that a data set wherein the 5 least similar images (of the 50) are the only ones of class y_q should not be treated the same as one wherein the 5 most similar images are the ones of class y_q.

6.2. Hyperparameters

There were multiple hyperparameters in the algorithm. The first was the split of the data set. We decided to do a 1:9 split because 10% of the data set was the right amount



(a) Non black pixel count = 0 (b) Non black pixel count = 0 (c) Non black pixel count = 117



(d) Non black pixel count = 212 (e) Non black pixel count = 224 (f) Non black pixel count = 371

Figure 6: Non black pixel count for some potential query images.

of training instances for the model to learn enough to have imperfect knowledge with considerable room for improvements.

The threshold value to distinguish between background and object pixels will also heavily affect the number of pixels counted as non black pixels. Obviously, depending on the threshold value the α value can be adjusted. However, for simplicity we decided to use 0.5 because any value above 0.5 is more white than it is black allowing us to clas-

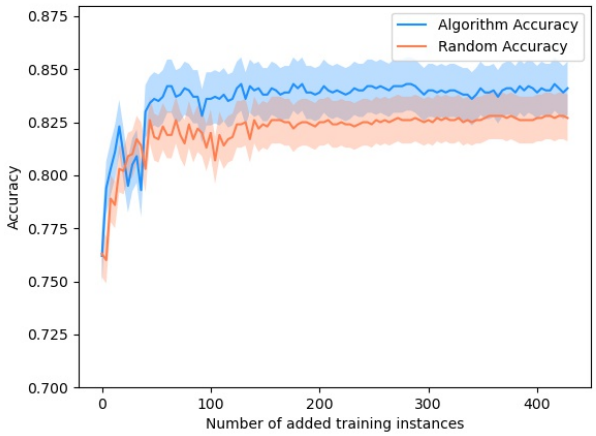


Figure 7: Results of trials averaged. Note: Some trials added more images than over (all added lesser than $maxAdditions$ though). We did not exclude that data.

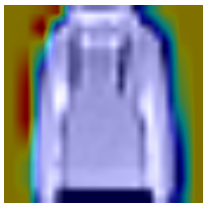


Figure 8: Example of the grad-CAM output revealing focus on object edges.

sify it as a light gray pixel and the reverse argument applies for dark gray pixels. Using a threshold of 0 is not advised because we have noticed that there are many potential query images for which the edges of the object have been identified as the most important regions by the model (refer to Figure 8). For these images the edges are surrounded by mostly dark gray pixels which will counted in the non black pixel count even though this is an example of misidentification of features. We defined α as 25% of the image thus the count much be $0.25 \times (28 \times 28) = 196$ which we rounded off to 200. 28×28 is the dimension of the FMNIST data set.

References

[1] A. Adadi and M. Berrada. Peeking inside the black-box: A survey on explainable artificial intelligence (xai). *IEEE Access*, 6:52138–52160, 2018. 2

[2] A. Barredo Arrieta, N. Díaz-Rodríguez, J. Del Ser, A. Bennetot, S. Tabik, A. Barbado, S. Garcia, S. Gil-Lopez, D. Molina, R. Benjamins, R. Chatila, and F. Herrera. Explainable artificial intelligence (xai): Concepts, taxonomies, opportunities and challenges toward responsible ai. *Information Fusion*, 58:82–115, 2020. 2

[3] K. Burns, L. A. Hendricks, T. Darrell, and A. Rohrbach. Women also snowboard: Overcoming bias in captioning models. *CoRR*, abs/1803.09797, 2018. 2

[4] A. Das and P. Rad. Opportunities and challenges in explainable artificial intelligence (XAI): A survey. *CoRR*, abs/2006.11371, 2020. 2

[5] S. Dasgupta, D. Hsu, and C. Monteleoni. A general agnostic active learning algorithm. Technical report, 2007. 1

[6] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009. 1

[7] D. Doran, S. Schulz, and T. R. Besold. What does explainable AI really mean? A new conceptualization of perspectives. *CoRR*, abs/1710.00794, 2017. 2

[8] Y. Gal, R. Islam, and Z. Ghahramani. Deep Bayesian Active Learning with Image Data. *arXiv:1703.02910 [cs, stat]*, Mar. 2017. arXiv: 1703.02910. 1, 2

[9] W.-N. Hsu and H.-T. Lin. Active Learning by Learning. page 7. 2

[10] S. Kaufman, S. Rosset, and C. Perlich. Leakage in data mining: Formulation, detection, and avoidance. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '11, page 556–563, New York, NY, USA, 2011. Association for Computing Machinery. 2

[11] D. D. Lewis and J. Catlett. Heterogeneous uncertainty sampling for supervised learning. In *In Proceedings of the Eleventh International Conference on Machine Learning*, pages 148–156. Morgan Kaufmann, 1994. 2

[12] D. D. Lewis and W. A. Gale. A sequential algorithm for training text classifiers. *CoRR*, abs/cmp-lg/9407020, 1994. 2

[13] N. Mayer, E. Ilg, P. Hausser, P. Fischer, D. Cremers, A. Dosovitskiy, and T. Brox. A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016. 1

[14] I. Mondal and D. Ganguly. ALEX: Active Learning based Enhancement of a Model's Explainability. *arXiv:2009.00859 [cs]*, Sept. 2020. arXiv: 2009.00859. 1, 2

[15] M. T. Ribeiro, S. Singh, and C. Guestrin. "Why Should I Trust You?": Explaining the Predictions of Any Classifier. *arXiv:1602.04938 [cs, stat]*, Aug. 2016. arXiv: 1602.04938. 2

[16] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra. Grad-CAM: Visual Explanations From Deep Networks via Gradient-Based Localization. page 9. 2, 3

[17] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, Oct 2017. 3

[18] R. R. Selvaraju, A. Das, R. Vedantam, M. Cogswell, D. Parikh, and D. Batra. Grad-cam: Why did you say that?, 2017. 3

[19] B. Settles. Active learning literature survey. Computer Sciences Technical Report 1648, University of Wisconsin–Madison, 2009. 1, 2

[20] C. Szegedy, A. Toshev, and D. Erhan. Deep Neural Networks for Object Detection. page 9. 1

[21] J. Wu, V. S. Sheng, J. Zhang, H. Li, T. Dadakova, C. L. Swisher, Z. Cui, and P. Zhao. Multi-label active learning algorithms for image classification: Overview and future promise. *ACM Comput. Surv.*, 53(2), Mar. 2020. 2

[22] H. Xiao, K. Rasul, and R. Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *CoRR*, abs/1708.07747, 2017. 4

[23] Q. Zhang. Convolutional Neural Networks. page 6, 2018. 4