XAI-Guided Interventional Retraining for Domain Generalization

2020-04-02

Suppose m is a model trained in one domain. Here is a simple method to obtain, from m, another model m' that hopefully generalizes better to other domains.

Let $\{h_u\}_{u=1}^F$ be all the feature units, and $\{z_c\}_{c=1}^C$ be the logit units. The weight between h_u and z_c in m is w_{uc}.

- For each training example $(\mathbf{x}_i, y_i)$, feed it to the model m and run an XAI method to obtain an heatmap e_i, which is what m considers as the core evidence in $\mathbf{x}_i$ for the class y_i. Assume e_i is normalized to the interval $[0, 1]$.

- Create a "purified" input $\mathbf{x}'_i$ by combining $\mathbf{x}_i$ and e_i. The simplest way is to do pointwise multiplication. We can also convert e_i into a binary mask using a threshold before the pointwise multiplication. (CWOX+Grad-CAM should be better than Grad-CAM in identifying the core evidence for the output.)

- Feed $\mathbf{x}'_i$ to m and compute the activation h'_{ui} of each feature unit h_u. Let $a_{ui} = h'_{ui} w_{uy_i}$. It is the contribution of feature h_u for classifying the purified example $\mathbf{x}'_i$ into the class y_i.

  Assume e_i correctly highlights the core evidence in $\mathbf{x}_i$ for class y_i. Then the features h_u that do not focus on the core evidence would have low activation h'_{ui} on the "purified" example $\mathbf{x}'_i$, and hence a_{ui} would be low. Also, a_{ui} would be small when w_{uy_i} is small. As such, a_{ui} would be high only if: (1) the feature h_u focus on the core evidence for y_i, and (2) it is important for the class y_i according to the model m.

  If a feature h_u is important for y_i according to m (i.e., with high w_{uy_i}), and it does not focus on the core evidence, it would lead to poor domain generalizability (DG). For better DG, we want to reduce the impact of such features. This can be achieved if we reduce the impact of all features with low a_{ui}. Note that by doing this we are also reducing the impact features with low w_{uy_i}, which should not bring about much impacts because the weights are low already.

- For each class c, define

$$A_{uc} = \frac{\sum_{i:y_i=c} a_{ui}}{\sum_{i:y_i=c} 1}$$

  Note that $\sum_{i:y_i=c} 1$ is the number of examples in the class c in the training set. So, A_{uc} is the average of the a_{ui}'s for the training examples in the class c. [A_{uc} can be computed on a subset of examples with low C-perplexity, although the retraining is done on all examples.]

Let say that the feature units h_u with $A_{uc} \geq \delta$ are *important* to the class c and the others are *unimportant* to c. Here, δ is a hyperparameter.

- To obtain m',
  - Set the weights for the features unimportant to the class c to 0, and keep them frozen during the following retraining process. The weights in the backbone are also frozen.
  - Re-initialize the weights for the important features, and retrain them.

This method feels more "heavy-handed" than previous methods. It is expected to lead to a decrease of the iid test error, and hopefully also lead to an increase of the ood test error.

2020-03-31

Suppose m is a model trained in one domain. Here is a simple method to obtain, from m, another model m' that hopefully generalize better to other domains.

Let $\{h_i\}_{i=1}^F$ be all the feature units, and $\{z_j\}_{j=1}^C$ be the logit units. The weight between h_i and z_j in m is w_{ij}.

How important is the unit h_i to a class j relative to all other classes $j' \neq j$ in m? This relative importance can be measured using

$$r_{ij} = \frac{e^{w_{ij}}}{\sum_{j'} e^{w_{ij'}}} \text{ or } r_{ij} = \frac{w_{ij}}{\sum_{j'} |w_{ij'}|} \text{or something else, e.g., add temperature in the first method}$$

For each class j, pick the top k features h_i with the largest r_{ij}, and regard them as *important features* for class j. The other units are *unimportant features* for class j.

`To obtain` m'`, set the weights between unimportant units and each class` j `to 0, and retrain the weights between the important units and each class. The parameters in other parts of the model` m `are frozen during the retraining.`

If a feature h_i contains information for multiple classes, it might be regarded as unimportant to any class although their weights might be large. The retraining will force the model not to use such features, and hence might increase domain generalizability to some extent.

When h_i is important only for one class j, it might still contain some background information that is spuriously correlated with the class j. This might limit the improvement on domain generalizability. *Can we use some ideas from XAI to identify features not containing spurious background information?* Here is a very rough idea: For each training example $(\mathbf{x}_n, y_n)$, use the result of some XAI method to "purify" $\mathbf{x}_n$ to get $\mathbf{x}'_n$. Figure out the "attribution" of each h_i when classifying $\mathbf{x}'_n$ to class y_n. Get a each for each h_i by somehow aggregating the attribution scores for all $n \in \{1, N\}$.

In Gao Han's work, we determine whether a feature unit h_i important for class j by check if it is important when classifying purified examples of class j. Such features should not have much spurious background information, and hence lead to good domain generalizability.

2020-03

Earlier, we talked about a method to retrain a model by changing the latent representations of inputs so as to focus on the latent features that correspond to the "core" part of the inputs. We expect the retrained model will generalize better to new domains.

There might be more direct ways to do this. We start with a dataset $\{\mathbf{x}_i, y_i\}_{i=1}^{N}$ and train a model m on it. m can be a model pretrained by others.

Then we select of subset of training examples $\{\mathbf{x}_i, y_i\}_{i=1}^{N_1}$ using X/C-perplexity. For each example $(\mathbf{x}_i, y_i)$ in the subset, we create a new example $(\mathbf{x}'_i, y_i)$, where $\mathbf{x}'_i$ is obtained from $\mathbf{x}_i$ be **removing** the pixels not covered by the heatmap for y_i. **Removing** a pixel might be:

1. Set it to 0, or
2. Set it to a random value. In this case, we can create multiple instances of $\mathbf{x}'_i$ for each $\mathbf{x}_i$, which might help.

Lastly, we fine-tune the model m using either of the two datasets

1. $\{\mathbf{x}'_i, y_i\}_{i=1}^{N_1}$, or
2. $\{\mathbf{x}_i, y_i\}_{i=1}^{N} \cup \{\mathbf{x}'_i, y_i\}_{i=1}^{N_1}$

During fine-tuning, the model needs to learn to classify the inputs based on the core parts, and hence should generalize better to new domains. Another perspective: Spurious correlations are removed in $\mathbf{x}'_i$. So, the method can be understood as a data augmentation method for remove spurious correlations.

In addition, during fine-tuning we can freeze parameters of the earlier layers, and fine-tune only those at the last l layers, where l can be 1 or a small number.

2020-02

Models trained in one domain usually do not perform well in other domains. For example, models trained on ImageNet have much lower accuracies on ImageNet V2. *Domain adaptation* is about learning models that generalize well across domains.

In this note, I describe a method for domain adaptation that consists of two training phases: (1) regular training, and (2) interventional retraining (IR). When classifying an input image, the output depends on (1) the object(s) of interest in the image, and (2) the background. The purpose of IR is to reduce the impact of the background. It is guided by XAI. So, we call it *XAI-Guided Interventional Retraining (XGIR).*

Let $\{\mathbf{x}_i, y_i\}_{i=1}^{N}$ be our training data. During regular training, we obtain a feature mapping $h = f_\theta(\mathbf{x})$, which converts the data into $\{h_i, y_i\}_{i=1}^{N}$. We also obtain a Softmax model $P(y|h, w)$. XGIR will change only w, but not θ. It does so by converting $\{h_i, y_i\}_{i=1}^{N}$

into $\{h'_i, y_i\}_{i=1}^N$ and then retrain a softmax model on the new data.

Given $f_\theta(\mathbf{x})$, $P(y|h, w)$ and $\{\mathbf{x}_i, y_i\}$, we obtain h'_i as follows:

- Run CWOX to get a XAI heatmap $\mathbf{e}_i$ for y_i
  - It could be the heatmap for the class y_i itself (See Figure 1 (b.1))
  - Or, the heatmap for the confusion cluster that contains y_i (See Figure 1 (c))
- $\mathbf{x}'_i = \mathbf{x}_i \odot \mathit{supp}(\mathbf{e}_i, \delta_1)$, where δ_1 is a small hyper-parameter, e.g., 0.01. After this step, some of the background is removed. Some foreground is also removed. Hope it does not matter much. Maybe $\delta_1 = \max_p \mathbf{e}_i(p) \times 0.15$, where p stands for pixel.
- $\mathbf{x}'_i$ is the **core** part of $\mathbf{x}_i$ that contains the evidence for the class y_i. It is the part the a good model should consider when classifying $\mathbf{x}_i$ into class y_i.
- Let $\mathbf{r}_i = f_\theta(\mathbf{x}'_i) - f_\theta(\mathbf{x}^0))$, where $\mathbf{x}^0$ is the empty image. $\mathbf{r}_i$ is a vector over the units of the feature layer. Ideally, the classifier should relying those units with high values $\mathbf{r}_i(u)$.
- We **intervene** to change feature values $h_i(u)$ for units with low $\mathbf{r}_i(u)$ via **randomization**:

  $$h'_i(u) = (1 - \lambda_i(u))h_i(u) + \lambda_i(u)\epsilon_i,$$

  where ϵ_i is random noise generated from a Gaussian distribution, and

  $$\lambda_i(u) = \frac{1}{\delta_2}[\delta_2 - \min\{\delta_2, \mathbf{r}_i(u)\}].$$

  The hyper-parameter δ_2 determines which units are considered to be the core units that capture the information that y_i should rely on. Maybe $\delta_2 = \max_u \mathbf{r}_i(u) \times 0.5$, where p stands for pixel.

  In fact, if $\mathbf{r}_i(u) \geq \delta_2$, than $h'_i(u) = h_i(u)$. The feature is not changed as all. If $\mathbf{r}_i(u) = 0$, on the other hand, $h'_i(u) = \epsilon_i$ is a purely random value. **The model needs to learn not to rely on it**.
- Do the retraining on a subset of examples with low C-perplexity and X-Perplexity

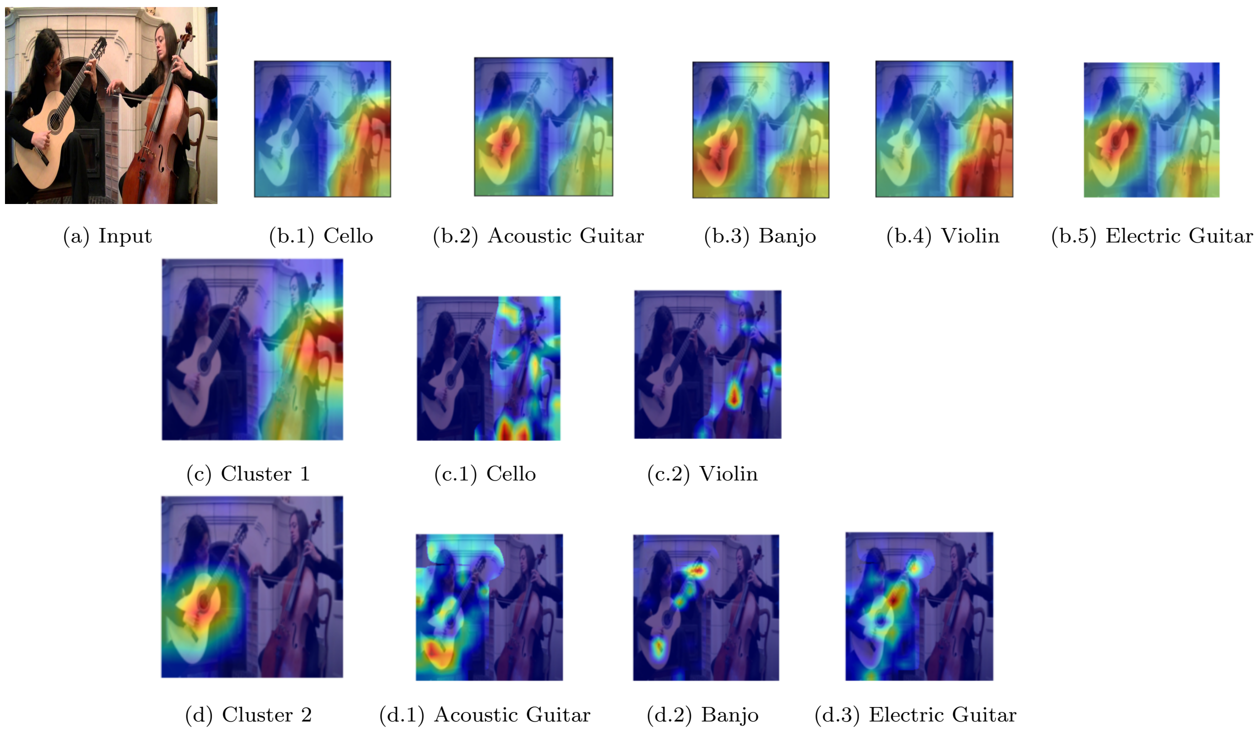

Figure 1: Individual output explanation (IOX), simple whole-output explanation (SWOX), and contrastive whole-output explanation (CWOX): (a) Input image with ground-truth label `cello`; (b.1) Grad-CAM heatmap for the top class (IOX); (b.1) - (b.5) Grad-CAM heatmaps for all 5 top classes (SWOX); (c) - (d) CWOX: Contrastive heatmaps are generated to first contrast the two confusion clusters (c, d), and then to contrast the classes in each confusion cluster against each other (c.1, c.2; d.1, d.2, d.3).