XAI-Guided Interventional Retraining for Domain Adaptation

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.

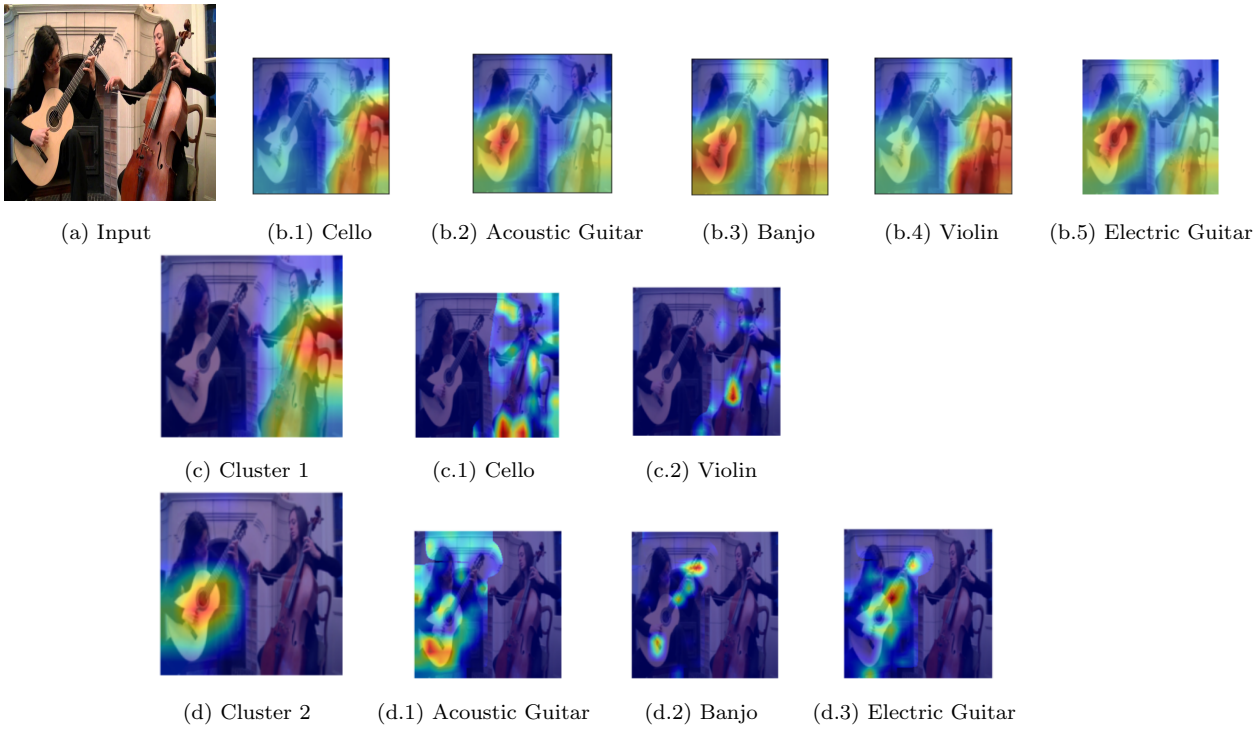
Given $f_\theta(\mathbf{x})$, $P(y|h, w)$ and $\{\mathbf{x}_i, y_i\}$, we obtain h'_i as follows:

- 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))
- $\mathbf{x}'_i = \mathbf{x}_i \odot supp(\mathbf{e}_i, \delta_1)$, 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.

  $\mathbf{x}'_i$ 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



(a) Input (b.1) Cello (b.2) Acoustic Guitar (b.3) Banjo (b.4) Violin (b.5) Electric Guitar

(c) Cluster 1 (c.1) Cello (c.2) Violin

(d) Cluster 2 (d.1) Acoustic Guitar (d.2) Banjo (d.3) Electric Guitar

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).

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

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.

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**.

In the project, try the idea on the setup used in the Two-Stage Classification project.