Objective to optimize

$$\Omega^A(\alpha_A, \alpha_P, \theta)$$
$$= \min_{\alpha_A \alpha_P \theta_1 \theta_2} (\alpha_A r_A + \alpha_P r_P) + \frac{1}{n_A n_P} \sum_{(i,j) \in M} \big(\text{CrossEntropy}(\text{Softmax}(g(x_j^P, a_i^A, \theta)), y_j^P) + \max(g(x_j^P, a_i^A, \theta)_{y_j^P} - \alpha_P \kappa_P, 0)$$
$$- \alpha_A ||x_i^A - x_j^P||\big)$$
$$\Omega^P(\alpha_A, \alpha_P, \theta)$$
$$= \min_{\alpha_A \alpha_P \theta_1 \theta_2} (\alpha_A r_A + \alpha_P r_P) + \frac{1}{n_A n_P} \sum_{(i,j) \in M} \big(\text{CrossEntropy}(\text{Softmax}(g(x_i^A, a_i^A, \theta)), y_j^P) + \max(g(x_i^A, a_i^A, \theta)_{y_j^P} - \alpha_P \kappa_P, 0)$$
$$- \alpha_P ||x_i^A - x_j^P||\big)$$

Constrained by

$$C^A = \{(\alpha_A, \alpha_P, \theta) : ||\theta_1||_* \leq \alpha_A + \alpha_P, ||\theta_2|| \leq \kappa_A \alpha_A, \alpha_A < \alpha_P\}$$
$$C^P = \{(\alpha_A, \alpha_P, \theta) : ||\theta_1||_* \leq \alpha_A + \alpha_P, ||\theta_2|| \leq \kappa_A \alpha_A, \alpha_A > \alpha_P\}$$

Notes

1. Pytorch has softmax embedded in CrossEntropyLoss already, so implementation doesn't require softmax layer on the output logits explicitly.
2. θ_1 is being interpreted as the classifier layer's weights in our setup. θ_2 is not being utilized yet because we don't have auxiliary feature for now.

Ongoing Challenges

1. The norm term $\alpha_A ||x_i^A - x_j^P||$ is on a larger scale than the general loss. Since α_A is also a parameter to optimize over, it's being taken advantage of a lot.

   A temporary solution right now is to put a sigmoid function to scale it down. But this leaves the question on how much we care about each term and the fundamental changes this scaling causes.

2. Another problem with the norm term $\alpha_A ||x_i^A - x_j^P||$ is that x_i and x_j as raw input are fundamentally too different as images. The intuition is that the more different (larger norm difference) x_i and x_j are, the less loss incurred so as to encourage robust classification. However, images are fundamentally too different. It's not easy to incorporate this intuition while the norm term is already on a larger scale and x_i, x_j are also fundamentally different regardless of their conceptual similarity.

   The temporary solution right now is to use the predicted logits to incorporate the similarity. Instead of using $\alpha_A ||x_i^A - x_j^P||$, we use $\text{Sigmoid}(\alpha_A ||g(x_i^A, \theta) - g(x_j^P, \theta)||)$ where sigmoid is used to scale down the loss incurred.

3. Right now, the model fails to learn and tend to **predict the same class for all samples**