Algorithm 1 Graph-MDAN meta-training

```
Input:  Source domains $\mathcal{D}_{\mathcal{S}_i}$; target domain $\mathcal{D}_{\mathcal{T}}$;
        Step size hyperparameters $\alpha$, $\beta$; domain adaptation hyperparameters $\gamma$, $\mu$;
Output: Neural network $\{\theta_f, \theta_G, \theta_y\}$;
Randomly initialize model parameter $\theta_f, \theta_G, \theta_y, \theta_d$
while stopping criterion is not met do
    Sample minibatch $\{\mathbf{x}_j^{\mathcal{S}_i}, y_j^{\mathcal{S}_i}\}_{j=1}^m$, $\{\mathbf{x}_j^{\mathcal{T}}\}_{j=1}^m$ from sources and target $\{\mathcal{D}_{\mathcal{S}_i}\}_{i=1}^k, \mathcal{D}_{\mathcal{T}}$
    for $t$ in $0, \cdots, T$ steps do
        # Parameter updating via gradient descent
        Compute $\mathcal{L}_{adp} = \mathcal{L}_{cls}(\theta_f, \theta_G, \theta_y) + \gamma \mathcal{L}_{con}(\theta_f, \theta_G)$
        Compute graph consistency parameters $\theta_G^t = \theta_G^t - \alpha \nabla_{\theta_G} \mathcal{L}_{adp}$
        Compute label predictor parameters $\theta_y^t = \theta_y^t - \alpha \nabla_{\theta_y} \mathcal{L}_{cls}$
    end for
    # Parameter adaptation via meta-gradient
    Compute $\mathcal{L}_{cls}(\theta_f, \theta_G^*, \theta_y^*)$, $\mathcal{L}_{con}(\theta_f, \theta_G^*)$, $\mathcal{L}_{dsc}(\theta_f, \theta_G^*, \theta_d)$ with initial $\theta_G^*$, $\theta_y^*$
    Update $\theta_y = \theta_y - \beta \nabla_{\theta_y} \mathcal{L}_{cls}$, and $\theta_d = \theta_d - \beta \mu \nabla_{\theta_d} \mathcal{L}_{dsc}$
    Update $\theta_G = \theta_G - \beta(\nabla_{\theta_G} \mathcal{L}_{cls} + \gamma \nabla_{\theta_G} \mathcal{L}_{con} - \mu \nabla_{\theta_G} \mathcal{L}_{dsc})$
    Update $\theta_f = \theta_f - \beta(\nabla_{\theta_f} \mathcal{L}_{cls} + \gamma \nabla_{\theta_f} \mathcal{L}_{con} - \mu \nabla_{\theta_f} \mathcal{L}_{dsc})$
end while
```

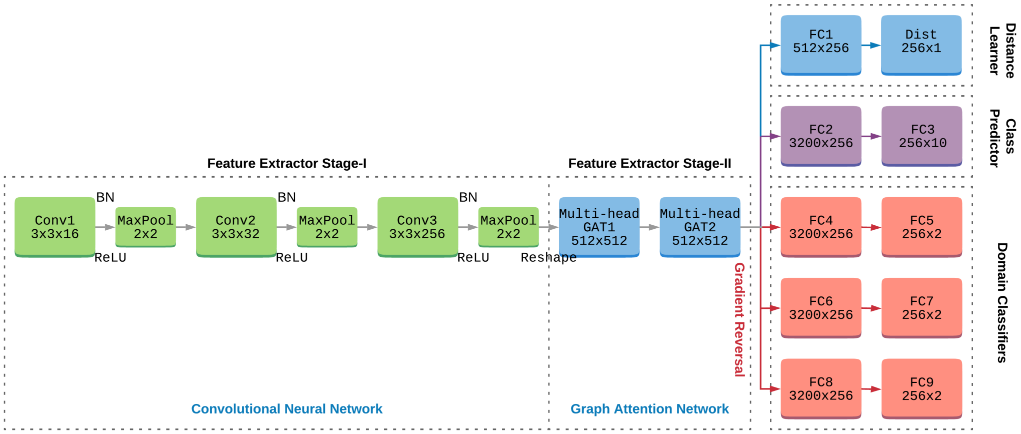

Figure 4: Graph-MDAN network architecture for digit classification.

Algorithm 2 Graph-MDAN meta-testing

```
Input:  Source domains $\mathcal{D}_{\mathcal{S}_i}$; target domain $\mathcal{D}_{\mathcal{T}}$; Hyperparameter $\alpha$, $\gamma$
        Learned parameter $\{\theta_f, \theta_G, \theta_y\}$ for the desired task;
Output: Neural network $\{\theta_f, \theta_G^*, \theta_y^*\}$;
Sampling testing minibatch $\{\mathbf{x}_j^{\mathcal{S}_i}, y_j^{\mathcal{S}_i}\}_{j=1}^m$, with the held-out $\{\mathbf{x}_j^{\mathcal{T}}\}_{j=1}^m$
Freeze feature extractor parameters $\theta_f$
# Parameter fast adaption with gradient descent:
Compute $\mathcal{L}_{adp} = \mathcal{L}_{cls}(\theta_f, \theta_G, \theta_y) + \gamma\mathcal{L}_{con}(\theta_f, \theta_G)$
Compute graph consistency parameters $\theta_G^* = \theta_G - \alpha\nabla_{\theta_G}\mathcal{L}_{adp}$
Compute label predictor parameters $\theta_y^* = \theta_y - \alpha\nabla_{\theta_y}\mathcal{L}_{cls}$
```