Algorithm 1 Layered Adaptive Differential Privacy (LADP)

Require: Model M, dataset D, total privacy budget ε, δ, number of epochs E

Ensure: Private model M'

for epoch $= 1$ to E **do**
    for batch B in D **do**
        $S \leftarrow$ EvaluateLayerSensitivities(M)
        $\varepsilon_{\text{layers}} \leftarrow$ AllocatePrivacyBudget(ε, S)
        $G \leftarrow$ ComputeGradients(M, B)
        for layer l in M.layers **do**
            if IsSensitiveLayer(l) **then**
                clip_norm $\leftarrow 1.0$
                noise_scale $\leftarrow \varepsilon_{\text{layers}}[l]$
            else
                clip_norm $\leftarrow 5.0$
                noise_scale $\leftarrow \varepsilon_{\text{layers}}[l] \times 0.5$
            end if
            $G[l] \leftarrow$ ClipGradient($G[l]$, clip_norm)
            $G[l] \leftarrow$ AddNoise($G[l]$, noise_scale)
        end for
        $M \leftarrow$ UpdateParameters(M, G)
        $\varepsilon \leftarrow \varepsilon - \sum \varepsilon_{\text{layers}}$
        if $\varepsilon \leq 0$ **then**
            return M
        end if
    end for
end for
return M
procedure EvaluateLayerSensitivities(M)
    // Use Fisher Information Matrix or Influence Functions
    // to evaluate sensitivity of each layer
    return layer_sensitivities
end procedure
procedure AllocatePrivacyBudget(ε, S)
    // Dynamically allocate privacy budget based on layer sensitivities
    // Use exponential mechanism for optimal allocation
    return layer_budgets
end procedure
procedure ClipGradient(gradient, clip_norm)
    return clip(gradient, −clip_norm, clip_norm)
end procedure
procedure AddNoise(gradient, noise_scale)
    noise $\leftarrow$ GenerateGaussianNoise(scale = noise_scale, shape = gradient.shape)
    return gradient + noise
end procedure