Algorithm 1: The **RMSProp** algorithm in [1]

Input α, γ: initial learning rate, decay rate
Input $\mu \in [0,1)$: parameters to control the momentum
Input $L(\theta)$: loss function with respect to the parameter θ
Input θ_0,v_0: initial parameter vector, and square average
$t \leftarrow 0$ (Initialize time step)
for t=1,2,…,T_e **do**
  $g_t \leftarrow \nabla L(\theta_{t-1})$ Get gradients at time step t
  $v_t \leftarrow \gamma v_{t-1} + (1-\gamma) g_t^2$ Get square average at time step t
  $\theta_t \leftarrow \theta_{t-1} - \alpha_t g_t / (\sqrt{\hat{v}_t} + \varepsilon)$ Update parameters
end for
Return θ_t (Trained parameters of DNNs)

Algorithm 2: The updated **RMSProp** algorithm **with** the TriOpts

Input α, γ: initial learning rate, decay rate
Input $\mu \in [0,1)$: parameters to control the momentum
Input $L(\theta)$: loss function with respect to the parameter θ
Input θ_0,v_0: initial parameter vector, and square average
$t \leftarrow 0$ (Initialize time step)
for t=1,2,…,T_e **do**
  $\tilde{g}_t \leftarrow \nabla L(\theta_{t-1}) - \mu(\nabla L(\theta_{t-1}))$ Get centralized gradients at time step t
  $v_t \leftarrow \gamma v_{t-1} + (1-\gamma) \tilde{g}_t^2$ Get square average at time step t
  $\alpha_t \leftarrow \alpha_0 (1+\cos(\pi \times t / T_e)) / 2$ Get learning rate at time step t
  $\theta_t \leftarrow \theta_{t-1} - \alpha_t \tilde{g}_t / (\sqrt{\hat{v}_t} + \varepsilon) + \mathrm{rand}(-\tau, \tau)$ Update parameters
end for
Return θ_t (Trained parameters of DNNs)

Algorithm 3: The **Nadam** algorithm in [2]

Input α_0, φ : initial learning rate, momentum decay
Input β_1, $\beta_1 \in [0,1)$: parameters to control the first and second moments
Input $L(\theta)$: loss function with respect to the parameter θ
Input θ_0, m_0, v_0: initial parameter vector, first moment, and second moment
$t \leftarrow 0$ (Initialize time step)
for t=1,2,…,T_e **do**
  $g_t \leftarrow \nabla L(\theta_{t-1})$ Get gradients at time step t
  $m_t \leftarrow \beta_1 m_{t-1} + (1-\beta_1) g_t$ Get first moment at time step t
  $v_t \leftarrow \beta_2 v_{t-1} + (1-\beta_2) g_t^2$ Get second moment at time step t
  $\mu_t \leftarrow \beta_1 (1 - 0.5 \times 0.96^{t\varphi})$ Get hyper-parameter at time step t
  $\hat{m}_t \leftarrow \mu_{t+1} m_t / (1 - \prod_{i=1}^{t+1} \mu_i) + (1-\mu_t) g_t / (1 - \prod_{i=1}^{t} \mu_i)$ Get bias-corrected first moment at time step t
  $\hat{v}_t \leftarrow v_t / (1-\beta_2^t)$ Get bias-corrected second moment at time step t
  $\theta_t \leftarrow \theta_{t-1} - \alpha_0 \hat{m}_t / (\sqrt{\hat{v}_t} + \varepsilon)$ Update parameters
end for
Return θ_t (Trained parameters of DNNs)

Algorithm 4: The updated **Nadam** algorithm **with** the TriOpts

Input α_0, φ : initial learning rate, momentum decay
Input β_1, $\beta_1 \in [0,1)$: parameters to control the first and second moments
Input $L(\theta)$: loss function with respect to the parameter θ
Input θ_0, m_0, v_0: initial parameter vector, first moment, and second moment
$t \leftarrow 0$ (Initialize time step)
for t=1,2,…,T_e **do**
  $\tilde{g}_t \leftarrow \nabla L(\theta_{t-1}) - \mu(\nabla L(\theta_{t-1}))$ Get centralized gradients at time step t
  $m_t \leftarrow \beta_1 m_{t-1} + (1-\beta_1) \tilde{g}_t$ Get first moment at time step t
  $v_t \leftarrow \beta_2 v_{t-1} + (1-\beta_2) \tilde{g}_t^2$ Get second moment at time step t
  $\mu_t \leftarrow \beta_1 (1 - 0.5 \times 0.96^{t\varphi})$ Get hyper-parameter at time step t
  $\hat{m}_t \leftarrow \mu_{t+1} m_t / (1 - \prod_{i=1}^{t+1} \mu_i) + (1-\mu_t) g_t / (1 - \prod_{i=1}^{t} \mu_i)$ Get bias-corrected first moment at time step t
  $\hat{v}_t \leftarrow v_t / (1-\beta_2^t)$ Get bias-corrected second moment at time step t
  $\alpha_t \leftarrow \alpha_0 (1+\cos(\pi \times t / T_e)) / 2$ Get learning rate at time step t
  $\theta_t \leftarrow \theta_{t-1} - \alpha_t \hat{m}_t / (\sqrt{\hat{v}_t} + \varepsilon) + \mathrm{rand}(-\tau, \tau)$ Update parameters
end for
Return θ_t (Trained parameters of DNNs)

Algorithm 5: The **Adamax** algorithm

Input α_0, T_e, τ : initial learning rate, training epoch, and noise parameter
Input β_1, $\beta_1 \in [0,1)$: parameters to control the first and second moments
Input $L(\theta)$: loss function with respect to the parameter θ
Input θ_0, m_0, μ_0 : initial parameter vector, first moment, infinity norm
$t \leftarrow 0$ (Initialize time step)
for t=1,2,…,T_e **do**
  $g_t \leftarrow \nabla L(\theta_{t-1})$ Get gradients at time step t
  $m_t \leftarrow \beta_1 m_{t-1} + (1-\beta_1) g_t$ Get first moment at time step t
  $\mu_t \leftarrow \max(\beta_2 \mu_{t-1}, |g_t| + \varepsilon)$ Get infinity norm at time step t
  $\theta_t \leftarrow \theta_{t-1} - \alpha_0 \hat{m}_t / (\mu_t - \beta_1 \mu_t)$ Update parameters
end for
Return θ_t (Trained parameters of DNNs)

Algorithm 6: The updated **Adamax** algorithm **with** the TriOpts

Input α_0, T_e, τ : initial learning rate, training epoch, and noise parameter
Input β_1, $\beta_1 \in [0,1)$: parameters to control the first and second moments
Input $L(\theta)$: loss function with respect to the parameter θ
Input θ_0, m_0, μ_0 : initial parameter vector, first moment, infinity norm
$t \leftarrow 0$ (Initialize time step)
for t=1,2,…,T_e **do**
  $\tilde{g}_t \leftarrow \nabla L(\theta_{t-1}) - \mu(\nabla L(\theta_{t-1}))$ Get centralized gradients at time step t
  $m_t \leftarrow \beta_1 m_{t-1} + (1-\beta_1) \tilde{g}_t$ Get first moment at time step t
  $\mu_t \leftarrow \max(\beta_2 \mu_{t-1}, |\tilde{g}_t| + \varepsilon)$ Get infinity norm at time step t
  $\alpha_t \leftarrow \alpha_0 (1+\cos(\pi \times t / T_e)) / 2$ Get learning rate at time step t
  $\theta_t \leftarrow \theta_{t-1} - \alpha_t \hat{m}_t / (\mu_t - \beta_1 \mu_t) + \mathrm{rand}(-\tau, \tau)$ Update parameters
end for
Return θ_t (Trained parameters of DNNs)

References

[1] Graves, Alex. "Generating sequences with recurrent neural networks." arXiv preprint arXiv:1308.0850 (2013).
[2] Dozat, Timothy. "Incorporating nesterov momentum into adam." 4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016.
[3] Kingma, Diederik P., and Jimmy Ba. "Adam: A method for stochastic optimization." arXiv preprint arXiv:1412.6980 (2014).
[4] https://pytorch.org/docs/stable/generated/torch.optim.Adamax.html#torch.optim.Adamax