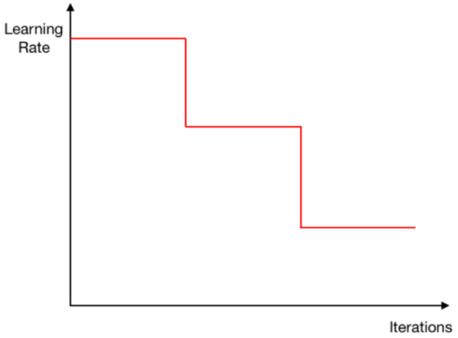
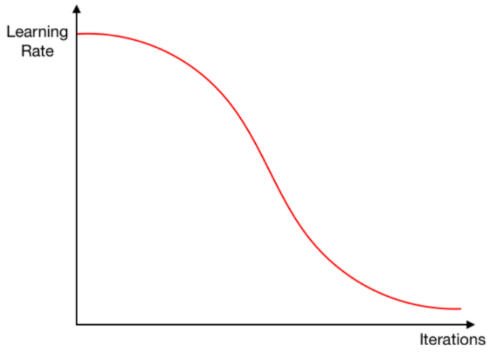
Learning Rate & AdamW

Learning Rate

- **Learning rate annealing**
  - 학습시 미리 정한 스케줄대로 learning rate를 바꿔가며 사용하는 것
  - 초기 learning rate를 상대적으로 크게 설정하여 Local minimum에 보다 더 빠르게 다가갈 수 있게 만들어주고 이후 learning rate를 줄여가며 local minimum에 보다 더 정확하게 수렴할 수 있게 함
  - 대표적인 예로 step function, cosine annealing이 존재

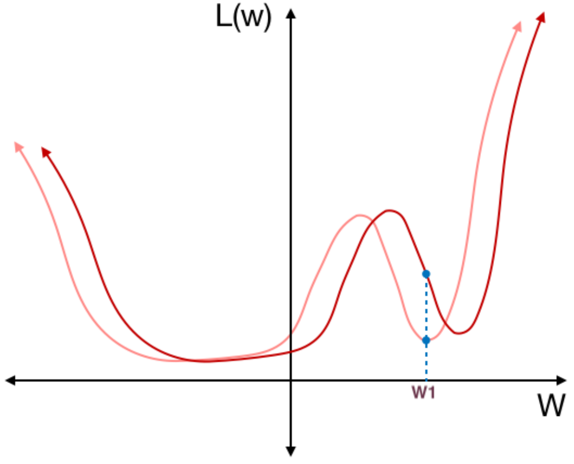




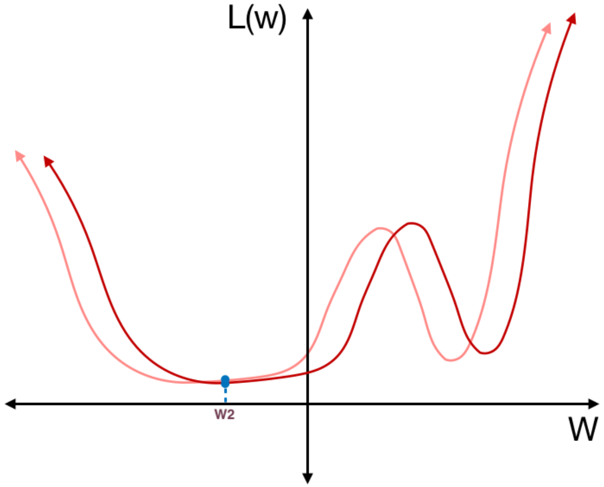
Learning Rate

- **Flat minima**
  - Loss Landscape에서 최종적으로 도달한 Minima가 Sharp한지 Flat한지에 따라 일반화 성능이 달라짐

  - 빨간 선이 train, 핑크 선이 test라 했을 때, 다른 분포의 데이터가 들어올 경우 flat한 지점이 상대적으로 안정적인 loss값을 가짐 => 보다 더 일반화 (generalized) 되었음



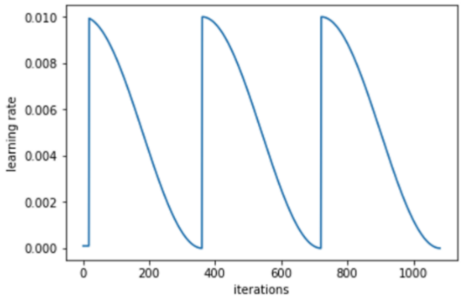
<sharp>



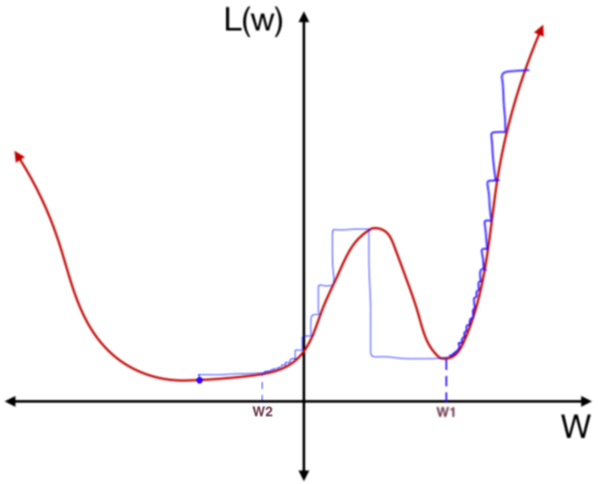
<flat>

Learning Rate

- **Warm restart**
  - Sharp minima에서 탈출해 Flat minima로 수렴하기 위해 고안된 방식



  - 학습 중간중간에 learning rate를 증가시켜 큰 폭의 weight update를 만들어 가파른 local minimum에서 빠져나올 기회를 제공
  - 주기적으로 learning rate를 증가시키면 그림과 같이 sharp minima에 탈출할 수 있게 되며, flat minima에서는 커지더라도 다시 회귀할 가능성이 높음



  - 이를 활용해 SAM, SWA와 같은 여러 기법들이 제안됨

AdamW

- AdamW는 (Loshchilov, I., & Hutter, F. (2017). Decoupled weight decay regularization. arXiv preprint arXiv:1711.05101.) 에서 처음 제안된 Optimizer

- 사전 연구를 통해 Adam이 momentum을 포함한 SGD에 비해 일반화(generalization)가 많이 뒤쳐진다는 결과 존재

- 해당 방식은 L2 regularization과 weight decay 관점에서 Adam이 SGD(momentum을 포함한)이 비해 일반화 능력이 떨어지는 이유를 설명

AdamW

- **L2 regularization**
  - L2 regularization은 손실함수에 weight에 대한 제곱텀을 추가해줘서 오버피팅을 방지해주는 방법
  - t번째 미니 배치에서의 손실 함수를 f_t, weight를 θ 라고 하면 L2 regularization을 포함한 손실함수 f_t^{reg}과 미분한 식은 다음과 같음

$$f_t^{reg}(\theta) = f_t(\theta) + \frac{\lambda'}{2}\sum_i \theta_i^2 \qquad \nabla f_t^{reg}(\theta) = \nabla f_t(\theta) + \lambda'\theta$$

- **Weight Decay**
  - weight decay는 기존의 gradient descent에서 weight 업데이트를 할 때, 이전 weight의 크기를 decay rate라는 0과 1 사이의 상수를 활용해 일정 비율 감소 시켜 오버피팅을 방지해주는 방법

$$\theta_{t+1} = (1-\lambda)\theta_t - \alpha\nabla f_t(\theta_t)$$

AdamW

- **L2 regularization == Weight Decay?**
  - L2 regularization과 weight decay는 같다는 것이 통념
  - 하지만 본 논문에서는 이 두 방식이 다르다는 것을 증명

① SGD 방식

- L2 regularization이 포함된 weight 업데이트 식과 f_t^{reg}로 편미분한 식은 다음과 같음
- 이를 weight 업데이트 식에 대입하면 다음과 같이 표현됨
- 이때, $\acute{\lambda} = \frac{\lambda}{\alpha}$면 L2 regularization은 정확히 weight decay와 같은 역할
- regularization 상수 λ'이 learning rate α에 dependent
- α 변경시 λ'은 최적의 파라미터가 아님

$$\theta_{t+1} = \theta_t - \alpha \nabla f_t^{reg}(\theta_t) \qquad \nabla f_t^{reg}(\theta) = \nabla f_t(\theta) + \lambda'\theta$$

<SGD L2 regularization>

$$\begin{aligned} \theta_{t+1} &= \theta_t - \alpha \nabla f_t^{reg}(\theta_t) \\ &= \theta_t - \alpha(\nabla f_t(\theta_t) + \lambda'\theta) \\ &= (1 - \alpha\lambda')\theta_t - \alpha \nabla f_t(\theta) \end{aligned}$$

<SGD weight decay>

② Adam 방식

- Adam은 gradient의 1차 모먼트 m_t와 2차 모먼트 v_t를 사용하고 weight마다 다른 learning rate를 적용
- 세타에 대한 식을 간단히 정리하고 편미분식을 대입하면 다음과 같음
- weight decay만 적용한 weight 업데이트 식은 다음과 같음
- 해당 식은 1차 모멘텀이 단위 행렬이 아닌 이상 같지 않다
- λ'앞에 1차 모멘텀이 붙기에 SGD 경우보다 일반화 능력이 떨어짐

$$\begin{aligned} m_t &= \beta_1 m_{t-1} + (1-\beta_1)\nabla f_t(\theta_t) \\ v_t &= \beta_2 v_{t-1} + (1-\beta_2)\nabla f_t(\theta_t)^2 \\ \hat{m}_t &= \frac{m_t}{1-\beta_1^t} \quad \text{(Bias correction)} \\ \hat{v}_t &= \frac{v_t}{1-\beta_2^t} \quad \text{(Bias correction)} \\ \theta_{t+1} &= \theta_t - \frac{\alpha}{\sqrt{\hat{v}_t + \epsilon}}\hat{m}_t \end{aligned}$$

$$\begin{aligned} \theta_{t+1} &= \theta_t - \alpha M_t \nabla f_t^{reg}(\theta_t) \\ &= \theta_t - \alpha M_t(\nabla f_t(\theta_t) + \lambda'\theta) \\ &= \theta_t - \alpha\lambda' M_t \theta - \alpha M_t \nabla f_t(\theta_t) \end{aligned}$$

<Adam L2 regularization >

$$\theta_{t+1} = (1-\lambda)\theta_t - \alpha M_t \nabla f_t(\theta)$$

<Adam weight decay>

AdamW

- **SGDW & AdamW**
  - 앞서 발견한 수식의 부조화를 해결하기 위해 L2 regularization과 별도로 weight decay를 위한 텀을 수식에 추가
  - 초록색 부분을 직접적으로 weight 업데이트 식에 추가 함으로써 decoupled weight decay도 생기게 변경

Algorithm 1 SGD with L_2 regularization and SGD with decoupled weight decay (SGDW), both with momentum

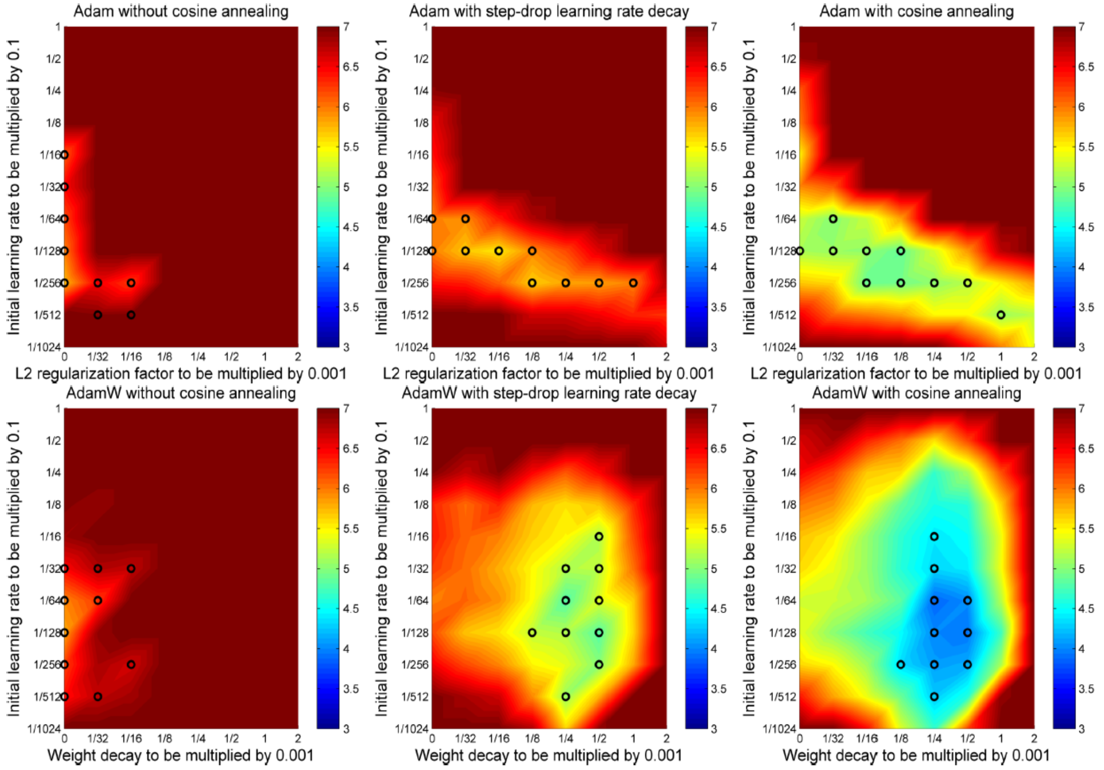
given initial learning rate $\alpha \in \mathbb{R}$, momentum factor $\beta_1 \in \mathbb{R}$, weight decay/L_2 regularization factor $\lambda \in \mathbb{R}$
initialize time step $t \leftarrow 0$, parameter vector $\boldsymbol{\theta}_{t=0} \in \mathbb{R}^n$, first moment vector $\boldsymbol{m}_{t=0} \leftarrow \boldsymbol{0}$, schedule multiplier $\eta_{t=0} \in \mathbb{R}$
repeat
  $t \leftarrow t + 1$
  $\nabla f_t(\boldsymbol{\theta}_{t-1}) \leftarrow \text{SelectBatch}(\boldsymbol{\theta}_{t-1})$ ▷ select batch and return the corresponding gradient
  $\boldsymbol{g}_t \leftarrow \nabla f_t(\boldsymbol{\theta}_{t-1})$ $+\lambda\boldsymbol{\theta}_{t-1}$
  $\eta_t \leftarrow \text{SetScheduleMultiplier}(t)$ ▷ can be fixed, decay, be used for warm restarts
  $\boldsymbol{m}_t \leftarrow \beta_1 \boldsymbol{m}_{t-1} + \eta_t \alpha \boldsymbol{g}_t$
  $\boldsymbol{\theta}_t \leftarrow \boldsymbol{\theta}_{t-1} - \boldsymbol{m}_t$ $-\eta_t \lambda \boldsymbol{\theta}_{t-1}$
until *stopping criterion is met*
return optimized parameters $\boldsymbol{\theta}_t$

Algorithm 2 Adam with L_2 regularization and Adam with decoupled weight decay (AdamW)

given $\alpha = 0.001, \beta_1 = 0.9, \beta_2 = 0.999, \epsilon = 10^{-8}, \lambda \in \mathbb{R}$
initialize time step $t \leftarrow 0$, parameter vector $\boldsymbol{\theta}_{t=0} \in \mathbb{R}^n$, first moment vector $\boldsymbol{m}_{t=0} \leftarrow \boldsymbol{0}$, second moment vector $\boldsymbol{v}_{t=0} \leftarrow \boldsymbol{0}$, schedule multiplier $\eta_{t=0} \in \mathbb{R}$
repeat
  $t \leftarrow t + 1$
  $\nabla f_t(\boldsymbol{\theta}_{t-1}) \leftarrow \text{SelectBatch}(\boldsymbol{\theta}_{t-1})$ ▷ select batch and return the corresponding gradient
  $\boldsymbol{g}_t \leftarrow \nabla f_t(\boldsymbol{\theta}_{t-1})$ $+\lambda\boldsymbol{\theta}_{t-1}$
  $\boldsymbol{m}_t \leftarrow \beta_1 \boldsymbol{m}_{t-1} + (1-\beta_1)\boldsymbol{g}_t$ ▷ here and below all operations are element-wise
  $\boldsymbol{v}_t \leftarrow \beta_2 \boldsymbol{v}_{t-1} + (1-\beta_2)\boldsymbol{g}_t^2$
  $\hat{\boldsymbol{m}}_t \leftarrow \boldsymbol{m}_t/(1-\beta_1^t)$ ▷ β_1 is taken to the power of t
  $\hat{\boldsymbol{v}}_t \leftarrow \boldsymbol{v}_t/(1-\beta_2^t)$ ▷ β_2 is taken to the power of t
  $\eta_t \leftarrow \text{SetScheduleMultiplier}(t)$ ▷ can be fixed, decay, or also be used for warm restarts
  $\boldsymbol{\theta}_t \leftarrow \boldsymbol{\theta}_{t-1} - \eta_t \left(\alpha \hat{\boldsymbol{m}}_t / (\sqrt{\hat{\boldsymbol{v}}_t} + \epsilon) + \lambda \boldsymbol{\theta}_{t-1} \right)$
until *stopping criterion is met*
return optimized parameters $\boldsymbol{\theta}_t$

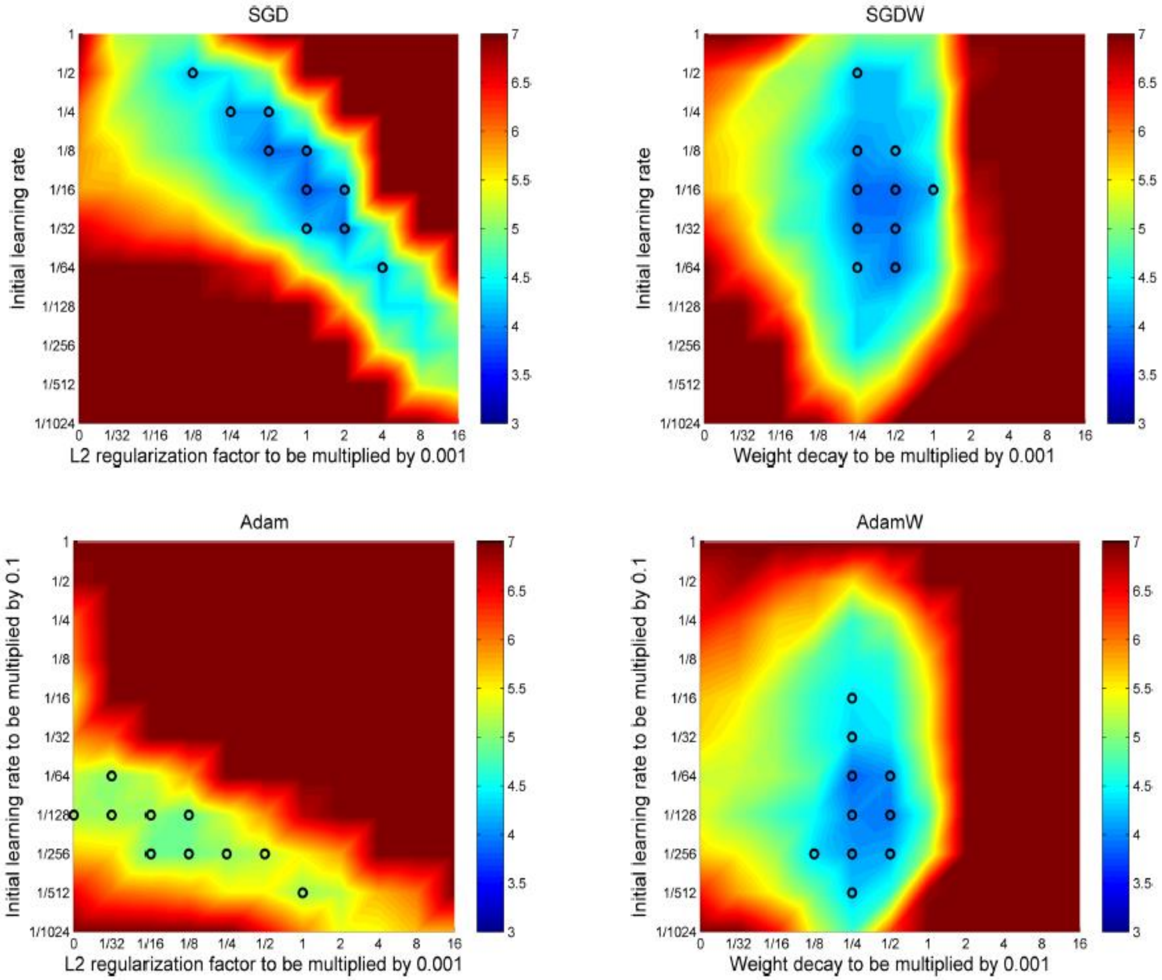
AdamW

- **Experiment Setting**
  - Dataset : CIFAR-10 / ImageNet의 다운샘플 버전
  - 모델 구조: Shake-Shake regularization 모델
- **Experiment1**
  - L2 regularization에 Adam을 적용했을 때와 weight decay까지 추가한 AdamW의 성능을 비교
  - 추가로 서로 다른 세 가지 lr schedule 에 대해서도 실험을 진행
  - AdamW가 Adam보다 모든 lr schedule에 대해 좋은 성능
  - 개인적으론 AdamW보다 lr schedule의 중요성을 강조하는 실험이라 생각이 듦



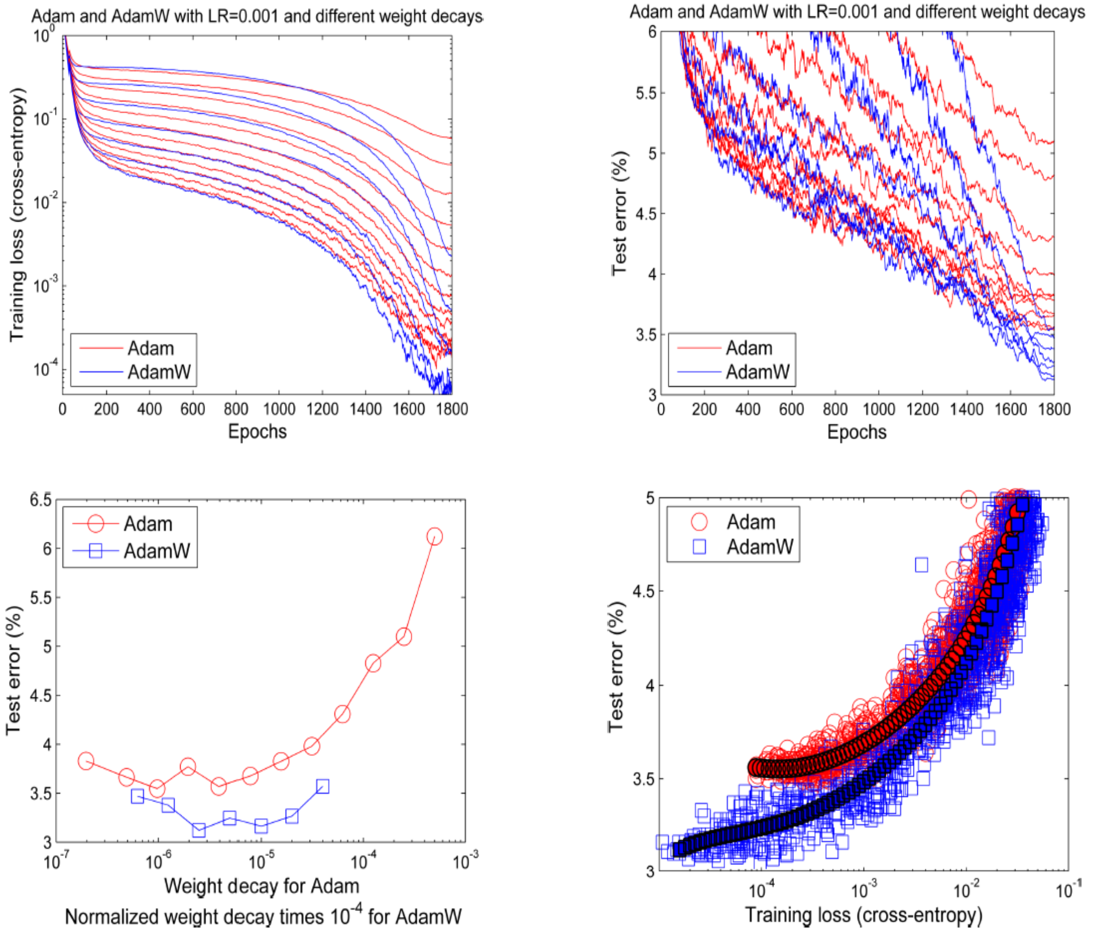
AdamW

- **Experiment2**
  - SGD vs SGDW , Adam vs AdamW을 각각 비교
  - 좌측에 존재하는 기존 optimizer는 weight decay 효과가 learning rate에 종속되어 강한 상관 관계를 보임
  - 이를 통해 기존 optimizer 는 최적의 하이퍼파라미터를 찾기 위해서는 α와 λ를 동시에 바꿔줘야 함
  - 반면 우측의 decoupled weight decay를 사용할 경우 learning rate와 weight decay가 서로 독립적
  - 본 논문에서 제안하는 optimizer 사용 시, 어느 한 하이퍼파라미터를 고정하고 다른 하나만을 바꿔가도 더 좋은 성능을 얻을 수 있음
  - AdamW는 SGD와 SGDW와 필적할 만한 성능



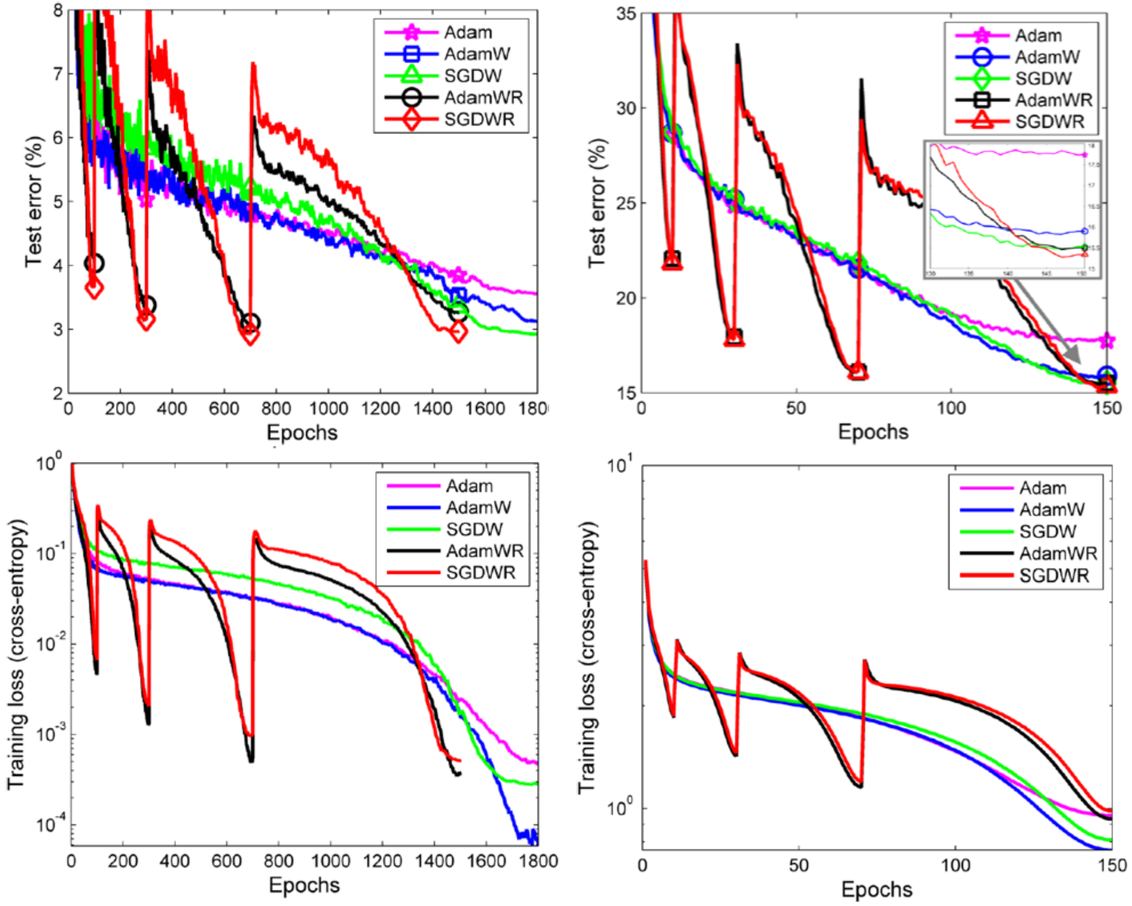
AdamW

- **Experiment3**
  - AdamW와 Adam의 일반화 능력 비교
  - 학습 초기에는 Adam과 AdamW과 비슷한 loss를 보이지만 학습이 진행될 수록 AdamW의 훈련 손실과 test 에러가 더 낮아짐



AdamW

- **Experiment4**
  - AdamWR vs SGDWR vs AdamW vs SGDW vs Adam
  - 좌측 : Epoch에 따른 Top-1 test error and training loss on CIFAR-10
  - 우측 : Epoch에 따른 Top-1 test error and training loss on ImageNet32x32
  - AdamWR과 SGDWR이 더 generalization을 잘한다는 것을 의미



AdamW

- **Conclusion**
  - Adaptive gradient methods 들은 L2 regularzation에 의한 weight decay 효과를 온전히 볼 수 없음
  - 이를 해결하기 위해 L2 regularziation에 의한 weight decay 효과와 별개로 weight decay를 weight 업데이트식에 넣음(Decoupled weight decay)
  - Learning rate schedule, 특히 cosine annealing이 Adam의 성능 상승에 도움을 줄 수 있다는 것을 확인
  - 또한 Warm restart도 성능 향상에 도움을 줄 수 있음

- **Conclusion**
  - 파이토치에서 공식으로 지원하는 optimizer

ADAMW

CLASS `torch.optim.AdamW(params, lr=0.001, betas=(0.9, 0.999), eps=1e-08, weight_decay=0.01, amsgrad=False, *, maximize=False, foreach=None, capturable=False)` [SOURCE]