

비트텐서(Bittensor)

해당 번역본은 이용자가 디지털자산 관련 정보를 편리하게 확인할 수 있도록
단순 참고용으로 번역된 것이며, 투자 권유를 목적으로 하지 않습니다.
그 내용이 정확하지 않을 수 있으므로 정확한 정보 습득을 위해서는
원문을 참고하시거나 원문 작성 주체 측에 문의하시기 바랍니다.
두나무는 해당 번역본 내용의 진실성 및 정확성을 보장하지 않고,
제공된 정보에 의한 투자 결과에 대하여 어떠한 책임도 지지 않으며,
투자 손실은 투자자에게 귀속됩니다.

비트텐서: P2P 인텔리전스 마켓

Yuma Rao

www.bittensor.com

초록

다른 상품과 마찬가지로, 시장은 인공지능(Machine Intelligence)을 효율적으로 생산할 수 있도록 도울 수 있습니다. 인텔리전스가 인터넷 전역의 다른 인텔리전스 시스템들에 의해 피어 투 피어(peer-to-peer) 방식으로 가격이 매겨지는 시장을 제안합니다. 피어들은 이웃의 가치를 학습하는 신경망을 훈련함으로써 서로를 순위화합니다. 점수는 디지털 원장(digital ledger)에 누적되며, 높은 순위를 차지한 피어들은 네트워크 내에서 추가적인 가중치로 금전적 보상을 받습니다. 그러나 이러한 형태의 피어 순위화는 담합(collusion)에 취약하며, 이는 메커니즘의 정확성을 방해할 수 있습니다. 그 해결책은 정직하게 선택된 가중치에 최대한의 보상을 부여하는 인센티브 메커니즘으로, 네트워크 전체 가중치의 최대 50%까지 담합에 저항할 수 있게 합니다. 그 결과, 집단적으로 운영되는 인텔리전스 시장이 지속적으로 새롭게 훈련된 모델을 생산하며, 정보 이론적 가치를 창출하는 기여자에게 보상을 지급하는 시스템이 됩니다.

인공지능(Machine Intelligence)의 생산은 거의 전적으로 벤치마킹(benchmarking) 시스템에 의존하게 되었습니다. 이 시스템에서는 머신 러닝 모델이 좁게 정의된 감독 학습(supervised) 문제에서 좋은 성능을 내도록 훈련됩니다. 이러한 시스템은 특정 문제에 대한 성능을 향상시키는 데에는 효과적이지만, 시장이 도입되면 훨씬 더 뛰어난 성과를 낼 수 있는 상황에서는 메커니즘이 약합니다. 예를 들어, 인텔리전스는 점점 더 특정 목표에서 분리되어, 다음과 같은 특성을 지닌 상품으로 변화하고 있습니다: (1) 데이터에서 고비용으로 채굴됨(Schwartz et al. [2019]), (2) 금전적 가치가 있음(OpenAI [2020]), (3) 이전 가능성(Devlin et al. [2019]), 그리고 (4) 일반적으로 유용함(Radford et al. [2019]). 감독 학습 목표로 그 생산을 측정하는 것은 상품 그 자체에 직접적인 보상을 제공하지 않으며, 이로 인해 분야가 좁은 전문화로 수렴하게 됩니다(Chollet [2019]). 게다가 이러한 목표는 종종 정확도와 같은 단일 차원의 지표로 측정되므로, 틈새 또는 기존 시스템에 대한 보상을 충분히 구분할 수 없습니다. 따라서 현재 최첨단이 아닌 것은 사라집니다. 궁극적으로 다양한 인텔리전스 시스템의 확산은 거대한 단일 모델을 훈련시켜 승자독식 구조에서 성공해야 한다는 필요성에 의해 제한됩니다. 독립적인 엔지니어들은 자신의 작업을 직접 수익화할 수 없으며, 그 결과 소수의 대기업이 최고의 인공지능에 대한 접근을 통제하는 중앙집중화가 발생합니다(OpenAI [2020]).

새로운 상품에는 새로운 형태의 시장이 필요합니다. 본 논문은 인공지능(Machine Intelligence)이 다른 인텔리전스 시스템에 의해 측정되는 프레임워크를 제시합니다. 모델은 훈련에 사용된 주관적 과제나 데이터셋과 무관하게 정보 생산 능력에 따라 순위가 매겨집니다. 인공지능(Machine Intelligence)을 측정하는 기준을 바꿈으로써, (1) 시장은 훨씬 더 다양한 목표에 적용 가능한 지능에 보상을 줄 수 있고, (2) 기존 시스템은 그 고유한 가치로 수익화될 수 있으며, (3) 더 작고 다양한 시스템은 훨씬 더 세밀한 보상 지형 내에서 틈새를 찾을 수 있습니다. 그 해결책은 인터넷을 통해 피어 투 피어(P2P) 방식으로 지속적이고 비동기적으로 서로 표현을 공유하는 컴퓨터 네트워크입니다. 구축된 시장은 디지털 원장을 사용하여 순위를 기록하고 피어들에게 탈중앙화된 방식으로 인센티브를 제공합니다. 블록체인은 신뢰를 측정하여, 피어가 다수에게 가치를 제공하지 않고는 보상을 얻기 어렵게 만듭니다. 연구자들은 인공지능(Machine Intelligence) 작업을 직접 수익화할 수 있고, 소비자들은 이를 직접 구매할 수 있습니다.

1. 구조

Hinton et al. [2015]의 정의를 따라, 데이터셋 $D = [X, Y]$ 위에서 손실 $\mathcal{L} = E_D[\mathcal{Q}(y, f(x))]$ 을 최소화하도록 훈련된 매개변수화된 함수 $y = f(x)$ 의 형태로 인텔리전스를 단순화하여 정의하는 것부터 시작합니다. 네트워크는 n 개의 함수 $F = f_0, f_1, f_2, \dots, f_n$ 으로 구성되며, 각 함수는 디지털 원장에 표현된 네트워크 가중치 $\mathbf{S} = [s_i]$ 또는 '스테이크(stake)'를 0개 이상 보유한 *피어(peer)*입니다. 이러한 함수들은 손실과 그들의 스테이크 비율과 함께 스테이크 가중 머신러닝 목표 $\sum_i^n \mathcal{L}_i * s_i$ 를 나타냅니다.

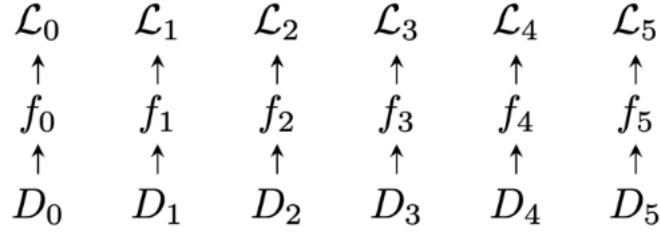
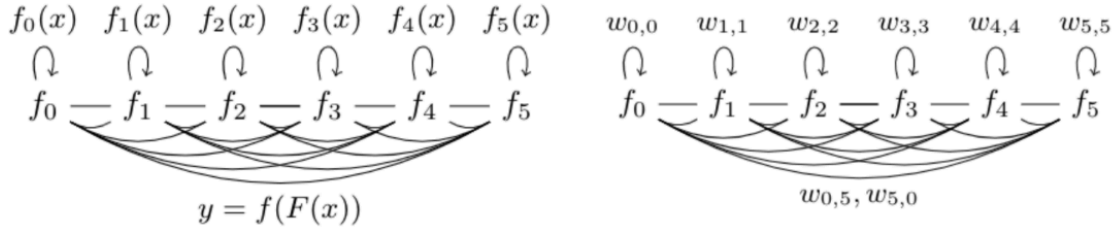


Figure 1: Peer functions with losses $\mathcal{L}_i$ and unique datasets D_i .

목표는 인센티브로서 스테이크 분배 I 를 손실 목표(Figure 1)를 최소화하는 데 기여한 피어들에게 지급하는 것입니다. 특히, 네트워크 내에서 손실을 최소화하지 않고 단지 분배를 극대화하기 위해 소수의 스테이크가 담합하는 것을 어렵게 하는 방식으로 이루어져야 합니다(Figure 3).

$$S_{t+1} = S_t + \tau I \quad (1)$$

본 논문에서는 이를 *피어 순위화(peer-ranking)*를 통해 달성할 수 있다고 제안합니다. 여기서 피어들은 다른 피어들의 출력 $F(x) = [f_0(x), f_1(x), \dots, f_n(x)]$ 을 자신의 입력 $f(F(x))$ 으로 사용하며, 가중치 집합 $\mathbf{W} = [w_{i,j}]$ 을 학습합니다. 이때 피어 i 는 디지털 원장 상의 트랜잭션을 통해 i 번째 행을 설정할 책임을 집니다.



(a) Inter-function connectivity.

(b) Weight matrix.

순위 계산에서 Fisher 정보 프루닝 점수(LeCun et al. [1989]; Yu et al. [2017])를 사용하여 가중치를 설정하면, $\mathbf{R} = \mathbf{W}^T \cdot \mathbf{S}$ 의 형태로 이 상적인 점수를 얻을 수 있습니다. 이때 각 피어의 인센티브는 그것을 네트워크에서 제거할 때 $\sum_i^n \mathcal{L}_i * s_i$ 에 유도되는 엔트로피 비용, 즉 *프루닝 점수(pruning score)*와 동등합니다.

$$r_i \approx \frac{1}{n} \sum_j \sum_{x \in D_j} \Delta F^T(x)_i \cdot \mathbf{H}(\mathcal{Q}_j(x)) \cdot \Delta F(x)_i \quad (2)$$

그러나 이 접근법은 담합에 대해 내성이 없습니다. 피어들이 (2)를 사용하는 대신 스스로에게 투표함으로써, 네트워크의 이익을 희생하고 자신들의 인플레이션을 증가시키도록 가중치를 설정할 수 있습니다(Figure 3). 이 공격은 디지털 원장이 각 모델의 파라미터를 감시할 수 없고, 오직 모델 간 가중치 $\mathbf{W}$ 만을 확인할 수 있기 때문에 매우 간단합니다.

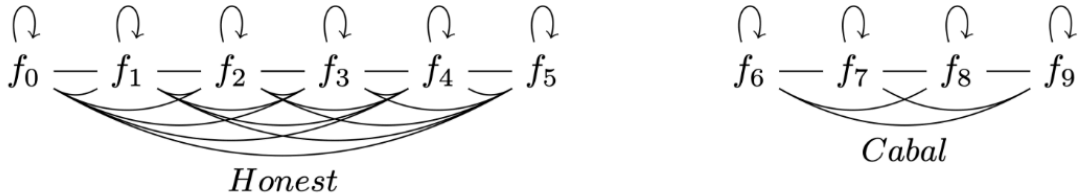


Figure 3: Disjoint cabal: peers in the right sub-network only vote for themselves.

2. 인센티브

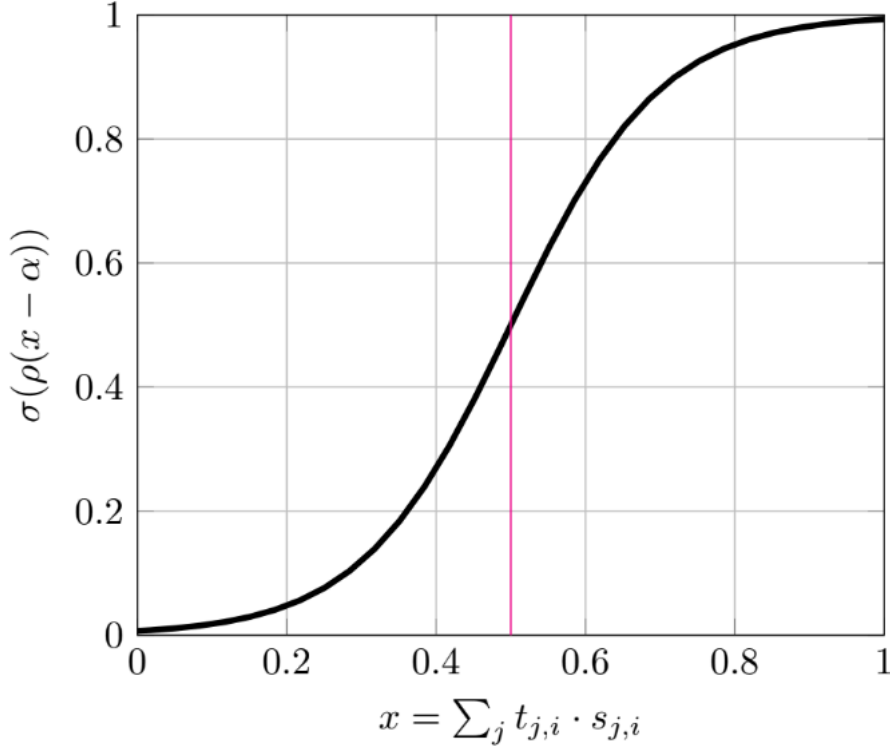
단순한 순위화 방법을 확장하여, 네트워크 내에서 합의(consensus)에 도달하지 않은 피어들에게만 보상이 제한되도록 하는 *인센티브(incentive)* 함수 $I(\mathbf{W}, \mathbf{S})$ 를 도입함으로써 담합(collusion)을 회피하도록 만들었습니다. 시스템 내 어떤 피어 그룹도 전체 스테이크(stake)의 과반수를 보유하지 않는다고 가정하면, 피어들은 다수의 지분으로부터 투표를 유도함으로써만 인플레이션을 얻을 수 있습니다. 이는 비트코인(Bitcoin)과 같은 많은 탈중앙화 시스템의 핵심 가정입니다.

용어를 다시 도입하면, 인센티브 메커니즘은 스테이크 벡터 $\mathbf{S}$ 와 피어 간 순위 행으로 구성된 가중치 집합 $\mathbf{W}$ 를 필요로 합니다. 또한 가중치로부터 순위 행렬 $\mathbf{T}$ 를 추론하며, 피어 i 와 j 사이에 제로가 아닌 엣지가 존재할 때 그리고 그때에만 $t_{i,j} = 1$ 이 됩니다.

$$\mathbf{W} = [w_{0,0} \ w_{0,1} \ w_{0,2} \ w_{0,3} \ w_{1,0} \ w_{1,1} \ w_{1,2} \ w_{1,3} \ w_{2,0} \ w_{2,1} \ w_{2,2} \ w_{2,3} \ w_{3,0} \ w_{3,1} \ w_{3,2} \ w_{3,3}] \quad \mathbf{S} = [s_0$$

합의(consensus)에 도달한 피어를, 네트워크 전체 스테이크의 50% 이상으로부터 제로가 아닌 엣지를 받은 피어로 정의합니다. (이는 단순히 정규화된 값 $(\mathbf{T}^T \cdot \mathbf{S}) > 0.5$ 로 표현됩니다.) 메커니즘이 미분 가능하도록 보장하기 위해, 이 계산을 연속적인 시그모이드(sigmoid) 함수를 사용하여 정의합니다. 시그모이드는 연결된 피어를 보상하고 신뢰받지 못한 피어를 벌하는 임계값(threshold) 기반의 스케일링을 생성합니다. 경사도와 임계점은 온도(temperature) ρ 및 시프트(shift) 항 κ 를 통해 조절할 수 있습니다.

$$\mathbf{C} = \sigma(\rho(\mathbf{T}^T \mathbf{S} - \kappa)) \quad (4)$$



합의 항을 사용하여 원래의 순위를 스케일링합니다. 피어들이 네트워크 내에서 더 많은 가중치를 획득함에 따라, 그들의 인플레이션은 0.5에 도달할 때까지 지수적으로 증가합니다. 10장에서 이 메커니즘이 두 경쟁 서브그래프 중 더 큰 쪽이 인플레이션을 통해 네트워크의 훨씬 더 큰 비율을 차지하게 함을 보여줍니다.

$$I = R \cdot C \quad (5)$$

3. 결속 (Bond)

위에서 설명한 합의 메커니즘은 작은 그룹이 인플레이션을 달성하기 어렵게 만들어 단순한 담합으로부터 보호합니다. 그러나 이는 올바른 가중치를 선택하는 데 대한 인센티브를 제공하지는 않습니다. bonds $\mathbf{B}$ 형태의 투기(speculation) 기반 보상을 인플레이션 메커니즘에 적용함으로써 이러한 인센티브를 도입합니다. 여기서 $b_{i,j} \in \mathbf{B}$ 는 피어 i 가 피어 j 에 보유한 결속의 비율을 의미합니다.

$$\mathbf{B} = [b_{0,0} \ b_{0,1} \ b_{0,2} \ b_{0,3} \ b_{1,0} \ b_{1,1} \ b_{1,2} \ b_{1,3} \ b_{2,0} \ b_{2,1} \ b_{2,2} \ b_{2,3} \ b_{3,0} \ b_{3,1} \ b_{3,2} \ b_{3,3}] \quad (6)$$

결속은 토큰 인플레이션과 유사하게 각 단계에서 누적됩니다. 즉, $\Delta \mathbf{B} = \mathbf{W} \cdot \mathbf{S}$ 입니다. 이러한 방식으로 피어들은 자신이 순위를 매긴 다른 피어들에 대한 결속을 축적하며, 결과적으로 자신을 그들과 결속(bonding) 시킵니다.

$$\mathbf{B}_{t+1} = \mathbf{B}_t + \mathbf{W} \cdot \mathbf{S} \quad (7)$$

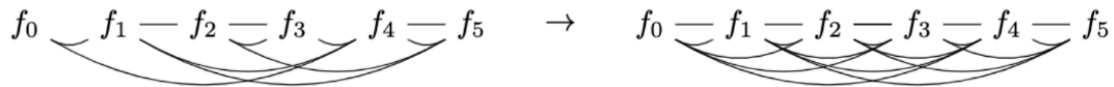
결속 행렬 $\mathbf{B}$ 를 사용하여, 블록체인은 일반적인 인센티브 점수를 $\Delta \mathbf{S} = \mathbf{B}^T \cdot \mathbf{I}$ 형태로 재분배합니다. 전통적인 주식 시장의 투기와 마찬가지로, 나중에 다른 피어들이 가치를 부여할 피어들에게 결속을 축적한 피어들은 자신도 더 높은 인플레이션을 얻게 됩니다. 따라서 피어들은 시스템 내에서 스테이크를 가진 다른 피어들의 판단에 따라 성과가 좋을 것으로 예상되는 피어에게 결속을 축적하는 것이 합리적입니다. 즉, 그들의 미래 가치에 대해 투기하는 것입니다. 마지막으로, 피어들이 자신들의 인플레이션 중 일정 비율을 확정적으로 얻을 수 있도록 메커니즘을 약간 수정합니다. 예를 들어, 50%의 비율을 적용하면 $\Delta \mathbf{S} = 0.5 \mathbf{B}^T \mathbf{I} + 0.5 \mathbf{I}$ 가 됩니다. $\Delta \mathbf{S}$ 는 n 개의 피어 전체에서 네트워크 인센티브를 결정하는 메커니즘의 단계별 업데이트가 됩니다.

$$S_{t+1} = S_t + \tau \Delta S \quad (8)$$

4. 합의 도달

2장에서 제시된 인센티브 함수는 신뢰도가 높은 피어에게 보상을 제공하지만, 정직한 노드들이 합의(consensus)에 도달하지 못할 경우 담합(collusion) 문제를 해결하지 못할 수도 있습니다. 특히 사용되지 않거나 잘못 설정된 스테이크(stake) 또는 부정확한 가중치 설정은 정직한 피어의 인플레이션 비율을 담합 서브네트워크(sub-network)에 비해 감소시킬 수 있습니다. 정직한 네트워크가 더 많은 스테이크를 보유하고 있더라도, 상대방을 압도할 만큼 충분한 인플레이션을 얻지 못할 수 있습니다. 반면 부정직한 서브그래프는 네트워크를 완전히 장악하지 않더라도, 가장 큰 경쟁자와 맞설 만큼의 인플레이션만 확보하면 됩니다.

이 공격은 토큰 인플레이션의 대부분이 그래프 내에서 과반수 신뢰를 받지 못한 피어들에게 분배될 때 발생할 수 있습니다. 체인은 이를 손실 항(loss term) $\mathcal{L} = -R \cdot (C - 0.5)$ 를 통해 측정할 수 있습니다(Figure 7). 인플레이션의 대부분이 합의 0.5 이상을 달성한 피어들에게 분배될 경우, 이 항은 음수가 됩니다. 체인은 이 손실 계산을 기준점(peg)으로 사용합니다. 네트워크 전반에 걸쳐 채굴자들이 설정하는 가중치의 수를 늘림으로써, 체인은 합의를 보장할 수 있습니다.



5. 네트워크 운영

네트워크 내에서 피어를 실행하는 단계는 다음과 같습니다:

1. 피어는 자신의 데이터셋 (D_i), 손실 함수 ($\mathcal{L}_i$), 그리고 매개변수화된 함수 (f_i)를 정의합니다.
2. 각 훈련 반복(iteration)에서, 피어는 자신의 데이터셋 (D_i)에서 샘플링한 배치(batch) 데이터를 조건적으로 다른 피어들에게 브로드캐스트합니다. 입력 형태는 ($x = [\text{batch_size}, \text{sequence_length}, \text{input_size}]$)입니다.
3. 응답 ($F(x) = [\dots f_j(x) \dots]$)—각각의 공통 형태 ($f_j(x) = [\text{batchsize}, \text{sequencelength}, \text{outputsize}]$)—는 게이팅 함수(gating function)를 통해 결합되어 로컬 모델 (f_i)의 입력으로 사용됩니다.
4. 타겟 레이블(target labels)과의 비교를 통해 손실 그라디언트(loss-gradient) ($\frac{\partial \mathcal{L}}{\partial F}$)가 생성되며, 이는 (f_i)를 거쳐 네트워크로 역전파됩니다(back-propagated).
5. 단계 2와 3 동안, 피어들은 다른 피어가 생성한 신호의 가치를 측정함으로써 자신에게 할당된 행의 가중치 ($w_{i,j} \in \mathbf{W}$)를 학습합니다.
6. 특정 시점 (t)에서 참가자들은 가중치 변화 ($\Delta \mathbf{W}_i$)를 제출하여 순위 ($\mathbf{R}$), 인플레이션 ($\mathbf{I}$), 합의 항 ($\mathbf{C}$), 그리고 결속 분배 ($\delta \mathbf{B}$)를 업데이트합니다.
7. 체인은 '손실(loss)'을 측정하고, 필요할 경우 새로 발행된 스테이크를 결속 소유 비율에 따라 네트워크 ($\Delta \mathbf{S}$)에 분배합니다.

6. 텐서(Tensor) 표준화

다양한 모델 유형과 입력 유형이 상호작용하기 위해서는 입력과 출력의 공통 인코딩이 필요합니다. 텐서(tensor) 모달리티(modality)의 사용은 네트워크를 분리된(disjoint) 그래프로 구분하는 데 사용할 수 있습니다. 초기에는 TEXT라는 단일 모달리티로 네트워크를 시드(seed)할 수 있으며, 이후 IMAGE, SPEECH, TENSOR 모달리티를 포함하도록 확장할 수 있습니다.

궁극적으로 TEXT-IMAGE와 같은 모달리티 조합을 추가하여 네트워크를 다중 모달리티(multi-modality) 환경으로 확장할 수 있습니다. 모달리티 간 연결에 대한 인센티브는 2장에서 제안한 신뢰 스케일링(trust scaling) 방식과 통합될 수 있습니다.

결국 성공적인 모델은 모든 모달리티의 입력을 수용하고, 이를 유용한 표현(representation)으로 변환할 수 있어야 합니다. 일관성을 위해, 언어 및 이미지 모델이 생성하는 일반적인 텐서 형태와 유사한 표준 출력 형태 ($[\text{batchsize}, \text{sequencedim}, \text{outputdim}]$)를 네트워크 전체에서 사용할 수 있으며, 네트워크의 복잡도가 증가함에 따라 이 크기를 확장할 수 있습니다.

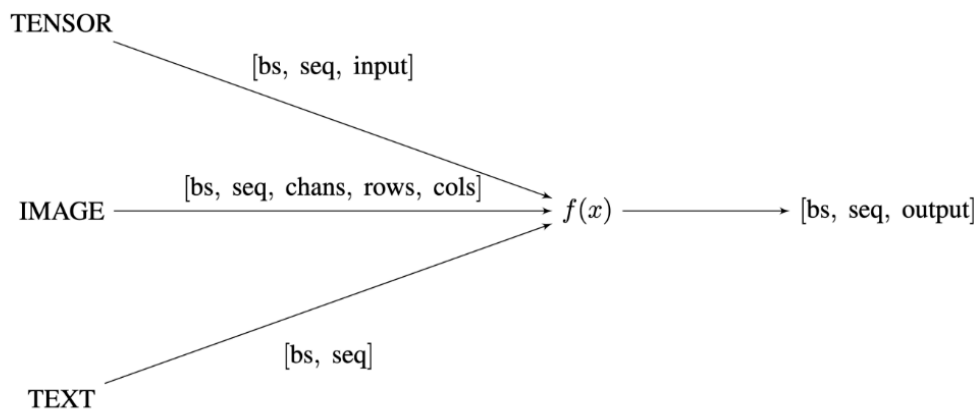


Figure 6: Standardization of input dimensions within the network

추상적인 입력 클래스에서 작업함으로써, 참가자들이 일반적인 멀티태스킹 학습 능력을 향상시키도록 할 수 있습니다(Kaiser et al. [2017]). 참가자들은 (1) 완전히 다른 컴퓨팅 기판(Nugent and Molter [2014]), (2) 데이터셋(Lample and Conneau [2019]), (3) 모델, (4) 그리고 시장 내에서 자신의 인센티브를 극대화하기 위한 전략을 사용할 수 있습니다. 데이터가 저렴하고 프라이버시가 요구되지 않는 비감독 학습(unsupervised) 데이터셋을 사용하는 것이 피어들에게 합리적입니다.

7. 조건부 연산

네트워크가 성장함에 따라, 외부 대역폭(outward bandwidth)은 주요 병목(bottleneck)이 될 가능성이 높습니다. 따라서 네트워크 전송을 줄이고 피어를 선택하는 방법이 필요합니다. 조건부 연산(conditional computation)은 피어들이 그래디언트 하강법(gradient descent)을 통해 네트워크 내에서 이웃을 선택하고 가지치기(pruning)하는 방법을 학습하도록 사용할 수 있습니다. 예를 들어, 제품 키 레이어(product key layer) 또는 희소 게이팅 레이어(sparsely gated layer)가 이에 해당합니다(Shazeer et al. [2017]).

$$f_i = f_i(G(x)) \quad (9)$$

$$G(x) = \sum_j g_j(x) * f_j(x) \quad (10)$$

조건부 레이어는 각 예시에 대해 질의할 희소한 피어 조합을 결정하고, 이를 곱셈적으로 결합함으로써 각 예시마다 소수의 피어만 질의하여 외부 대역폭을 절감합니다. 이 방법은 Shazeer et al. [2017], Ryabinin and Gusev [2020]이 보여준 바와 같이, 피어들이 그래프 내에서 훨씬 더 많은 이웃과 통신할 수 있도록 하여 외부 대역폭을 극적으로 증가시킬 수 있습니다. 본질적으로, 이 레이어는 입력에 기반하여 피어를 탐색하는 학습 가능한 DNS 조회 메커니즘으로 작동합니다. 또한 손실 함수에 대해 학습 가능하므로, 가중치 $w_{i,j} \in \mathbf{W}$ 에 대한 유용한 프록시(proxy)를 제공합니다.

8. 지식 추출

함수 간의 의존성은 모델들이 반드시 온라인 상태를 유지해야 하며, 프로덕션 환경에서 독립적으로 실행될 수 없음을 의미합니다. 이러한 의존성을 해소하기 위해 지식 증류(distillation) 기법을 사용할 수 있습니다(Hinton et al. [2015]). 이는 더 작은 모델(학생 모델, student)이 나머지 네트워크의 행동을 모방하도록 하는 압축 및 지식 추출 기술입니다. 증류 레이어(distillation layer)는 조건부 연산 계층(식 10)과 함께 사용되며, 학생 모델은 게이팅 네트워크(gating network)가 생성한 로짓(logit)과 학생의 예측 분포 간의 교차 엔트로피(여기서는 KL로 표시됨)를 통해 네트워크의 동작을 모방하도록 학습합니다(Sanh et al. [2020]).

$$\text{distillation loss} = KL_D(\text{dist}(x), G(x)) \quad (11)$$

증류된 모델은 네트워크의 대리(proxy)로 작동하므로, 모델들을 완전히 오프라인 상태로 전환하여 평가할 수 있습니다. 또한 네트워크 구성 요소 간의 재귀(recursion)가 차단되어 임의의 네트워크 그래프 구성이 가능해집니다. 모델이 오프라인으로 전환되더라도, 그 피어들은 증류된 버전을 대신 사용할 수 있습니다. 개인 데이터(private data)는 네트워크를 직접 질의하지 않고, 증류된 모델을 통해 검증할 수 있습니다. 궁극적으로, 구성 요소들은 증류된 모델을 활용하여 완전히 네트워크로부터 분리된 상태에서 검증 및 추론(inference)을 수행할 수 있습니다.

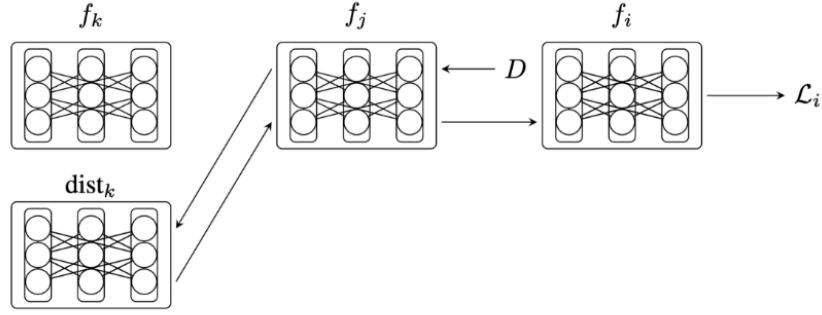


Figure 7: Queries propagate to depth=1 before the distilled model is used.

9. 가중치 학습

이 연구의 목표는 피어들 간의 순위 $r = [r_i]$ 를 생성하는 것입니다. 여기서 점수 $r_i \in \mathbf{R}$ 는 참가자의 벤치마크에 대한 정보 이론적 중요도 (information-theoretic significance)를 나타냅니다. LeCun et al. [1989]; Yu et al. [2017]의 접근에 따라, 이러한 중요도는 네트워크에서 각 피어를 제거할 때 발생하는 비용으로 정의하는 것이 타당합니다. 이 점수를 분석적으로 도출할 수 있으며, 여기서 $\Delta F(x)_i$ 는 네트워크에서 i^{th} 피어가 제거될 때 j^{th} 피어의 입력에 발생하는 교란(perturbation)을 의미합니다(부록 12.2 참조).

$$r_i \approx \frac{1}{n} \sum_j \sum_{x \in D_j} \Delta F^T(x)_i \cdot \mathbf{H}(\mathcal{Q}_j(x)) \cdot \Delta F(x)_i \quad (12)$$

$$\Delta F(x)_i = [0, \dots, 0, -f_i(x), 0, \dots, 0]$$

여기서 오류 함수 $\mathcal{Q}_j$ 가 2차 미분 가능한 교차 엔트로피(cross-entropy)일 경우, $\mathbf{H}(\mathcal{Q}_j)$ 는 Fisher 정보 행렬(Fisher-information matrix)이 되며, $r_i \in \mathbf{R}$ 는 각 피어의 네트워크 전체에 대한 정보적 중요도(informational significance)로 적절히 측정됩니다. 그러나 정보 이론적 가중치는 오류 함수의 전체 헤시안(Hessian)을 필요로 합니다. 실무적으로는 오류 함수로부터 입력으로 기여 점수를 전파(propagate)하는 휴리스틱을 사용하는 것이 더 현실적입니다(Yu et al. [2017]). 예를 들어, 6장에서 소개된 게이팅 레이어의 가중치는 유용한 미분 가능한 프록시(proxy)로 활용될 수 있습니다.

10. 담합(Collusion)

이제 네트워크 내 일부 피어들이 카발(cabal)을 형성하여, 이웃의 성능을 정확히 평가하지 않고도 자신들의 인플레이션을 극대화하려 시도하는 시나리오를 고려합니다. 정직한 그래프 A (스테이크 S_A 보유)와 분리된 카발 B (스테이크 S_B 보유) 간의 경쟁은 각자의 네트워크 스테이크 비율에 따라 결정됩니다. 정직한 그래프는 네트워크를 유지하고 보호하기 위해 더 많은 인플레이션을 확보해야 합니다. 즉, $I_A >> I_B$ 여야 합니다.

정직한 그래프의 스테이크 비율이 부정직한 그래프보다 크다고 가정합니다. 즉, $S_A > S_B$, 그리고 체인이 합의 상태($\mathcal{L} < 0$)에 도달했다고 가정합니다. 모든 B 의 피어들이 A 와 분리되어 있으므로, 손실 항 $-R_B \cdot (C_B - 0.5) > 0$ 은 양수가 됩니다. 반면 $\mathcal{L} < 0$ 이므로, $R_A \cdot (C_A - 0.5) < 0$ 은 음수가 되어 정직한 서브그래프 A 내에는 과반수와 연결된 피어들이 존재함을 의미합니다.

체인이 진행됨에 따라, 새로 발행된 스테이크는 인플레이션을 τ 에 비례하여 $I = R \cdot T$ 형태로 방출됩니다. 중요하게도, 인센티브 함수의 스테이크에 대한 그래디언트는 양수이며, 정직한 그래프와 부정직한 그래프의 변곡점에서 초선형(super-linear)입니다. 특히, $\frac{\partial I}{\partial S} = \frac{\rho}{2}$ 이므로, 각 서브그래프가 보유한 스테이크의 양은 다음 반복(iteration)에서 인플레이션의 비선형적 변화를 반영하게 됩니다.

초기에 $S_A > 0.5$ 이고 $S_B < 0.5$ 이므로, 서브그래프 A에서 방출되는 지분의 비율이 서브그래프 B에서 방출되는 비율보다 큼니다. 따라서 서브그래프 A의 인센티브는 B에 비해 초선형적으로(super-linearly) 증가합니다. 그 결과, 지분 비율 $\frac{S_B}{S_A + S_B}$ 은 감소하게 되며, 카발(cabal)은 시간의 흐름에 따라 자신을 유지하기 위해 지속적으로 자신의 서브그래프에 지분을 추가해야 합니다.

우리는 연속적인 인플레이션 하에서 경쟁하는 그래프들 간의 이 비율을 고려합니다. 이를 Python 코드로 변환하면 다음과 같습니다:

```
tau = 0.1
temp = 10
stake_A = 0.51
stake_B = 0.49
history = []
for block in range(100):
    total_stake = stake_A + stake_B
    trust_A = 1/(1 + math.exp(-(stake_A/total_stake - 0.5) * temp))
    trust_B = 1/(1 + math.exp(-(stake_B/total_stake - 0.5) * temp))
    ranks_A = stake_A
    ranks_B = stake_B
```

```

incentive_A = ranks_A * trust_A
incentive_B = ranks_B * trust_B
total_incentive = incentive_A + incentive_B
total_stake = stake_A + stake_B
stake_A += tau * total_stake * incentive_A / total_incentive
stake_B += tau * total_stake * incentive_B / total_incentive
print(block, stake_B / (stake_A + stake_B))

```

블록(block)	카발의 크기(size of cabal)
0	0.4877323388820201
1	0.4849535784321247
2	0.4815511535094221
3	0.4773899105039098
4	0.4723039486843246
5	0.46612224574620587
6	0.45861590847737577
7	0.44955887540065376
8	0.43871643745912897
9	0.42587900870651624
10	0.41090548935459825
...	...
90	0.0002827251010618101
91	0.00025719653886131316
92	0.0002339730247373799
93	0.000212846436856568
94	0.00019362744329658293
95	0.0001761438058611274
96	0.00016023883697158944
97	0.00014576999582759936
98	0.00013260761127280887
99	0.00012063371993464691

11. 결론

신뢰할 수 있는 환경(trusted environment) 외부의 P2P 네트워크 상에서 작동하는 지능 시장(intelligence market)을 제안하였습니다. 핵심적으로, 이 벤치마크는 다른 지능 시스템(intelligence systems)을 활용하여 표상적 지식 생성(representational-knowledge production) 성과를 측정함으로써 그 가치를 결정합니다. 이 과정이 협력적이고 고해상도로 수행될 수 있다는 사실은, 이 벤치마크가 전반적인 분야에서 더 나은 보상 메커니즘을 제공할 수 있음을 시사합니다.

이 목표를 달성하기 위해, 본 논문은 추상적으로 정의된 지능 모델들로 구성된 P2P 네트워크의 정의로부터 시작하였습니다. 이 프레임워크가 네트워크로부터 개별 피어(peer)를 제거하는 데 드는 비용을 기반으로 각 피어의 순위를 산출할 수 있게 함을 보였습니다. 피어들은 디지털 원장(digital ledger) 상의 가중치 집합을 이용해 이 점수를 상호 협상하였습니다. 그러나 참가자들이 부정직한 서브그래프(dishonest sub-graphs)를 형성하는 것을 방지하는 메커니즘이 없이는, 이 시스템은 불완전하다는 점이 드러났습니다.

이를 해결하기 위해, 우리는 피어 연결성(peer connectivity)에 기반한 인센티브 구조를 제안하였습니다. 이는 네트워크의 큰 부분으로부터 신뢰를 받는 피어들에게 지속적으로 보상을 제공하는 방식입니다. 이 메커니즘은 시간이 지남에 따라 부정직한 서버그래프들이 점차 무의미하게 쇠퇴하도록 보장합니다.

이후, 우리는 (1) 피어들이 미분 계층(differential layer)을 통해 연결성을 학습함으로써 네트워크 대역폭을 감소시키는 방법과, (2) 완전히 네트워크와 분리된 기계 학습 모델(machine learning models)을 추출하여 실제 환경에서 실행할 수 있는 방법을 제시하였습니다. 그 결과, 본 연구는 시스템 내 새로운 학습자들에게 지식을 생산하고 이를 제공하는 참가자들에게 보상을 부여하는 지능 시장(intelligence market)을 구현할 수 있음을 보였습니다.

참고문헌

R. Schwartz, J. Dodge, N. A. Smith, and O. Etzioni, “Green ai,” 2019.

OpenAI, “Openai licenses gpt-3 technology to microsoft,” *OpenAI Blog*, vol. 1, no. 1, p. 1, 2020.

J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, “Bert: Pre-training of deep bidirectional transformers for language understanding,” 2019.

A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever, “Language models are unsupervised multitask learners,” *OpenAI Blog*, vol. 1, no. 8, p. 9, 2019.

F. Chollet, “On the measure of intelligence,” 2019.

G. Hinton, O. Vinyals, and J. Dean, “Distilling the knowledge in a neural network,” 2015.

Y. LeCun, J. S. Denker, S. A. Solla, R. E. Howard, and L. D. Jackel, “Optimal brain damage.” in *NIPs*, vol. 2. Citeseer, 1989, pp. 598–605.

R. Yu, A. Li, C.-F. Chen, J.-H. Lai, V. I. Morariu, X. Han, M. Gao, C.-Y. Lin, and L. S. Davis, “Nisp: Pruning networks using neuron importance score propagation,” 2017.

L. Kaiser, A. N. Gomez, N. Shazeer, A. Vaswani, N. Parmar, L. Jones, and J. Uszkoreit, “One model to learn them all,” 2017.

M. A. Nugent and T. W. Wolter, “Cortical processing with thermodynamic-ram,” 2014.

G. Lample and A. Conneau, “Cross-lingual language model pretraining,” 2019.

N. Shazeer, A. Mirhoseini, K. Maziarz, A. Davis, Q. Le, G. Hinton, and J. Dean, “Outrageously large neural networks: The sparsely-gated mixture-of-experts layer,” 2017.

M. Ryabinin and A. Gusev, “Towards crowdsourced training of large neural networks using decentralized mixture-of-experts,” 2020.

V. Sanh, L. Debut, J. Chaumond, and T. Wolf, “Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter,” 2020.

D. Balduzzi, W. M. Czarnecki, T. W. Anthony, I. M. Gemp, E. Hughes, J. Z. Leibo, G. Piliouras, and T. Graepel, “Smooth markets: A basic mechanism for organizing gradient-based learners,” 2020.

P. Dütting, Z. Feng, H. Narasimhan, D. C. Parkes, and S. S. Ravindranath, “Optimal auctions through deep learning,” 2017.

12. 부록

12.1 순위 정확도

피어들이 체인 상의 가중치에 대해 자기 이익에 따라 업데이트를 수행할 때, 순위화 메커니즘의 정확도를 고려합니다. 피어들은 자신의 손실을 최소화하는 동시에 가장 큰 서버그래프 내에 연결된 상태를 유지하려는 인센티브 압력에 놓여 있습니다. 메커니즘이 충분히 수학적으로 구성되어 있으므로, 우리는 각 피어(peer)의 보상 함수(payload function)를 가중치(weight)의 미분 가능한 효용 함수(differentiable utility function)로 모델링합니다. 즉, 가중치는 다음과 같이 표현됩니다: $U(\mathcal{L}(\mathbf{W}))$

$$P_i(\mathbf{W}) = U_i(\mathcal{L}_i(\mathbf{W})) \quad (13)$$

피어의 효용 항(utility term) $U(\mathcal{L}(\mathbf{W}))$ 은 해당 피어가 자신의 손실을 최소화하려는 주관적 관심을 반영합니다. 이 피어를 다른 피어들과 연결하는 가중치의 크기를 줄이면 지식을 추출할 능력이 감소하므로, 효용 함수를 입력 변화로 모델링할 수 있습니다. 이 변화를 모델링하기 위해, 이동 임계값(shifted threshold)을 사용합니다. 즉, 피어 i 의 손실 함수 변화는 다른 피어들이 설정한 평균값보다 낮은 가중치에 해당하는 이웃의 입력이 마스크(masked)되는 것으로 반영됩니다. 이때 평균 가중치는 $\mu_j = \left(\frac{1}{n}\right) \sum_i s_i * w_{i,j}$ 로 정의됩니다.

$$F\mathbf{W}(x) = [f_0(x) * \sigma(s_i * w_{i,0} - \mu_0), \dots, f_n(x) * \sigma(s_i * w_{i,n} - \mu_n)] \quad (14)$$

$$\sigma = \frac{1}{1 + e^{-\frac{x}{\sigma}}} \quad (15)$$

가중치 변화에 따른 손실 변화를 도출하기 위해, 입력 섭동(input perturbation) ($F\mathbf{W} - F\mathbf{W}_0$)을 사용합니다. 여기서 $\mathbf{W}^0$ 는 초기 가중치 선택을 의미합니다. 7장에서 보았던 동일한 섭동 방식을 사용하여, 헤시안(Hessian) 항 $\mathbf{H}(\mathcal{L}(F))$ 하에서 손실 변화가 다음과 같이 계산됩니다 (12.3 참조):

$$\frac{\partial \mathcal{L}}{\partial \mathbf{W}} = \frac{\partial}{\partial \mathbf{W}} [(F\mathbf{W} - F\mathbf{W}_0)^T \cdot \mathbf{H}(\mathcal{L}(F)) \cdot (F\mathbf{W} - F\mathbf{W}_0)] \quad (16)$$

효용 함수에 대해 추가적인 선형 가정을 둡니다: $U_i(W) = \alpha \cdot \mathcal{L}_i(W)$. 이를 통해 피어의 효용에 대한 완전한 미분 가능 함수를 얻을 수 있습니다. 이 구성은 매끄러운 시장(smooth market)(Balduzzi et al. [2020])으로, 기울기 하강법(gradient descent)을 통해 경쟁 균형(competitive equilibrium)을 탐색할 수 있습니다. 이때 각 단계는 $\Delta \mathbf{W}_i = \frac{\partial P_i}{\partial \mathbf{W}_i}$ 로 정의됩니다.

$$\mathbf{W}_{t+1} = \mathbf{W}_t + \lambda \Delta \mathbf{W} \quad (17)$$

$$\Delta \mathbf{W} = \left[\frac{\partial P_0}{\partial \mathbf{W}_0} : \frac{\partial P_n}{\partial \mathbf{W}_n} \right] \quad (18)$$

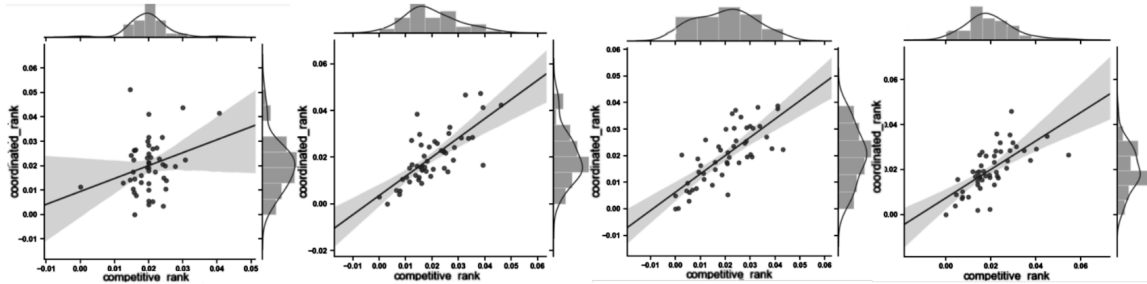
위의 경험적 모델로부터 통계를 생성하여 피어 순위화 방법의 정확도를 평가합니다. 먼저 메커니즘의 매개변수 $[\lambda, \alpha, n]$ 을 선택하고, 각 피어에 대해 n 개의 임의의 양의 준정부호(positive semi-definite) $n \times n$ 헤시안 항 $[\mathbf{H}]$ 을 포함하는 초기화된 네트워크 상태 $[\mathbf{W}^0, \mathbf{S}]$ 를 생성합니다. 초기화 이후, (16)에서 계산된 그래디언트 항을 사용하여 (18)의 하강 전략을 적용하고, 표준 그래디언트 하강 도구를 통해 시스템을 내재된 평형점으로 수렴시킵니다. 발견된 지역 최소값(local minimum)은 참여자들이 자신들의 가중치 선택을 변경하더라도 추가적인 이익을 얻을 수 없는 경쟁 균형(competitive equilibrium)을 의미합니다(Dütting et al. [2017]). 이 시점에서 경쟁 순위 $\mathbf{R}^*$ 를 계산하고, 이를 7장에서 논의된 헤시안으로부터 도출된 이상적 점수 $\mathbf{R}$ 과 비교합니다. 두 점수 간의 차이는 스피어만 상관 계수(Spearman's rho correlation)로 측정되며, 아래에 예시 실험 결과를 시각화합니다.

12.2 이상적인 순위 도출

벤치마크 $\mathcal{B} = \sum_i^n \mathcal{L}_i$ 의 변화를, 지역 최소점(local minimum) 근처에서 i^{th} 피어의 제거를 반영하는 섭동(perturbation) $\Delta F(x)_i = [\dots, -f_i(x), \dots]$ 하에 근사합니다.

$$\Delta \mathcal{B} = \mathcal{B}(F + \Delta F_i) - \mathcal{B}(F) = \sum_i [\mathcal{L}_i(F + \Delta F_i) - \mathcal{L}_i(F)] \quad (19)$$

$$\mathcal{L}_i(F + \Delta F_i) - \mathcal{L}_i(F) \approx \frac{\partial \mathcal{L}_i}{\partial F} \cdot \Delta F_i + \frac{1}{2} \Delta F_i^T \cdot \mathbf{H}(\mathcal{L}_i) \cdot \Delta F_i + O(\Delta F_i^3) + O(\Delta F_i^4) \quad (20)$$



식 (19)는 벤치마크의 정의로부터 유도되며, 식 (20)은 섭동 $\Delta F(x)_i$ 에 대한 테일러 급수(Taylor series)로부터 도출됩니다. 첫 번째 항 $\frac{\partial \mathcal{L}_i}{\partial F}$ 는 지역 최소점에서 0이 되며, 높은 차수 항 $O(\Delta F_i^3)$ 는 충분히 작은 섭동에 대해 무시할 수 있습니다. 이러한 가정은 LeCun et al. [1989] 및 Yu et al. [2017]에서도 동일하게 적용되었습니다. 또한, $\mathcal{L}_i$ 는 데이터셋 D_i 에 대한 기댓값(expectation)이므로, 모든 항은 특정 지점 x 에서 평가됩니다. 이에 따라 다음을 얻습니다:

$$\Delta \mathcal{B} \approx \frac{1}{2} \sum_i^n \sum_{x \in D_i} \Delta F^T(x) \cdot \mathbf{H}(\mathcal{Q}_i(x)) \cdot \Delta F_i(x) \quad (21)$$

여기서 오류 함수에 대한 헤시안(Hessian) $\mathbf{H}(\mathcal{Q} * i(x))$ 및 데이터셋에 대한 합 $\sum * x \in D_i$ 가 적절히 대체되었습니다. 상수 계수 $\frac{1}{2}$ 는 제거할 수 있으며, 이를 통해 최종 결과를 얻습니다.

12.3 가중치 수렴 게임 도출

12.4 정리(Theorem)

헤시안 선택 $\mathbf{H}(\mathcal{L}(F))$ 에 대해, 네트워크의 수렴 게임(convergence-game)은 다음과 같은 그래디언트 항 간의 선형 관계로 기술될 수 있습니다.

$$\frac{\partial P}{\partial \mathbf{W}} = \alpha \cdot \frac{\partial \mathcal{L}}{\partial \mathbf{W}} + \frac{\partial r}{\partial \mathbf{W}} \quad (22)$$

손실의 그래디언트는 다음과 같습니다:

$$\frac{\partial \mathcal{L}}{\partial \mathbf{W}} = \frac{\partial}{\partial \mathbf{W}} [(F\mathbf{W} - F\mathbf{W}_0)^T \cdot \mathbf{H}(\mathcal{L}(F)) \cdot (F\mathbf{W} - F\mathbf{W}_0)] \quad (23)$$

12.5 설정

참여자들의 행위를 두 가지 보상 항(payload term)을 통해 특성화하여 시스템을 분석합니다.

1. 각 참여자의 손실은 가중치의 함수로 표현되며, 그 효용(utility)은 $U(\mathcal{L}(\mathbf{W}))$ 로 주어집니다. U 는 가중치 행렬의 작은 변화에 대해 거의 선형(linear)이라고 가정하며, $U(\mathcal{L}) = \alpha * \mathcal{L}$ 로 표현됩니다. 따라서 $\frac{\partial U}{\partial \mathcal{L}} = \alpha$, 여기서 α 는 양수이며 0이 아닙니다.
2. 네트워크는 입력에 대해 로컬 최소점 상태로 수렴했으며, $\frac{\partial \mathcal{L}}{\partial F} = 0$ 입니다.

6.3절에서 정의한 보상 함수(payload formulation)를 사용하면 다음과 같이 쓸 수 있습니다.

$$P(\mathbf{W}) = \alpha \cdot \mathcal{L}(\mathbf{W}) + r(\mathbf{W}) \quad (24)$$

효용 함수와 방출(emission)은 유사한 단위로 측정되므로, α 는 손실의 단위 변화당 가격(price)을 의미합니다.

이 분석은 그러한 점수가 존재한다고 가정할 뿐, 실제 계산 가능성을 의미하지는 않습니다.

참여자들은 자신의 로컬 보상을 극대화하기 위해 그래디언트 단계 $\Delta \mathbf{W}_i = \frac{\partial P_i}{\partial \mathbf{W}_i}$ 를 따라 가중치를 선택합니다.

이후의 분석에서는 간결성을 위해 첨자 i 를 생략합니다. 입력의 변화 ΔF 에 대해 손실의 테일러 전개를 고려하면 다음과 같습니다.

$$\mathcal{L}(F + \Delta F) = \mathcal{L}(F) + \frac{\partial \mathcal{L}}{\partial F} \Delta F + \frac{1}{2} \Delta F \cdot \mathbf{H}(\mathcal{L}(F)) \cdot \Delta F + O(\Delta F^3) \quad (25)$$

첫 번째 선형 항 $\frac{\partial \mathcal{L}}{\partial F}$ 는 최소점에서 0이 되며, 충분히 작은 섭동 $\bar{F}$ 의 경우 고차항은 제거됩니다. 이제 변수 변환을 수행합니다: $F = F\mathbf{W}_0$, 그리고 $\Delta F = F\mathbf{W}_1 - F\mathbf{W}_0$. 여기서 $\mathbf{W}_0$ 는 최소점에서의 가중치이며, $\mathbf{W}_1$ 은 또 다른 선택으로서 $F\mathbf{W}_0$ 및 $F\mathbf{W}_1$ 은 각각 $\mathbf{W}_0$ 및 $\mathbf{W}_1$ 에 의해 마스크된 입력을 나타냅니다.

이를 식 (25)에 대입하면:

$$\mathcal{L}(F\mathbf{W}_1) = \mathcal{L}(F\mathbf{W}_0) + \frac{1}{2} (F\mathbf{W}_1 - F\mathbf{W}_0)^T \cdot \mathbf{H}(\mathcal{L}(F)) \cdot (F\mathbf{W}_1 - F\mathbf{W}_0) \quad (26)$$

함수 $\mathcal{L}(F\mathbf{W}_1)$ 은 네트워크가 이미 $\mathbf{W}_0$ 하에서 수렴했다고 가정할 때, 임의의 가중치 선택 $\mathbf{W}_1$ 에 대한 손실의 근사값입니다. 마지막으로, 효용 함수의 α -선형 가정을 적용하면 다음을 얻습니다.

$$\frac{\partial U}{\partial \mathbf{W}} = \alpha \cdot \frac{\partial \mathcal{L}}{\partial \mathbf{W}} \approx \frac{\alpha}{2} \frac{\partial}{\partial \mathbf{W}} [(F\mathbf{W} - F\mathbf{W}_0)^T \cdot \mathbf{H}(\mathcal{L}(F)) \cdot (F\mathbf{W} - F\mathbf{W}_0)] \quad (27)$$

간결성을 위해 첨자 $\mathbf{W}_1$ 은 생략했으며, $\mathcal{L}(F\mathbf{W}_0)$ 는 상수이므로 가중치 선택에 의존하지 않습니다. 또한 상수 계수 $\frac{1}{2}$ 는 미지의 α 에 포함시킬 수 있습니다. 남은 항 $\frac{\partial r}{\partial \mathbf{W}}$ 는 1장에서 제시된 순위 함수로부터 유도 가능합니다.

따라서 다음의 결과를 얻습니다.

$$\frac{\partial P}{\partial \mathbf{W}} \approx \alpha \cdot \frac{\partial \mathcal{L}}{\partial \mathbf{W}} + \frac{\partial R}{\partial \mathbf{W}} \quad (28)$$

$$\frac{\partial \mathcal{L}}{\partial \mathbf{W}} = \frac{\partial}{\partial \mathbf{W}} [(F\mathbf{W} - F\mathbf{W}_0)^T \cdot \mathbf{H}(\mathcal{L}(F)) \cdot (F\mathbf{W} - F\mathbf{W}_0)] \quad (29)$$

12.6 사후(ex-post) 무후회(zero-regret) 단계 도출

위에서 설명된 시스템을 고려한다. n 개의 피어 집합이 학습률 λ 를 사용한 경사 하강(gradient descent)으로 순위 행렬 $\mathbf{W}$ 의 가중치를 반복적으로 변경하고 있다. $\mathbf{W}^{t+1} = \mathbf{W}^t + \lambda \Delta \mathbf{W}$. 여기서 가중치 변화는 $\Delta \mathbf{W} = [\Delta w_0, \dots, \Delta w_n]$ 이며, 각 Δw_i 는 피어 i 가 푸시한 단일 행(row)에 대한 변화이다. 각 피어는 가중치의 함수인 자신의 보상 $P_i(\mathbf{W})$ 를 경쟁적으로 최대화하려 한다.

12.6.1 정의

단일 단계에 대한 사후(ex-post) 후회(regret)는 선택된 단계 Δw_i 와 모든 대안 Δw_i^* 사이의 손실 차이의 최댓값이다. 기대 사후 후회(expected ex-post regret)는 다른 참여자들이 선택한 모든 Δw_j 들에 대한 기대값으로 취한 이 차이를 의미한다(Dütting et al. [2017]).

$$rgt_i = E_{\Delta w_j} [\max_{\Delta w_i^*} [P_i(\Delta w_i^*) - P_i(\Delta w_i)]] \quad (30)$$

12.6.2 정리(Theorem)

충분히 작은 λ 에 대해, 전략 $\Delta w_i = \frac{\partial P}{\partial w_i}$ 의 기대 사후 후회는 0이다.

12.6.3 증명

가중치 변화 $\mathbf{W}^* = \mathbf{W} + \lambda \Delta \mathbf{W}$ 하에서 보상 함수 P 에 대해 점 $\mathbf{W}$ 에서의 테일러 정리를 고려한다. $\mathbf{W}^* \rightarrow \mathbf{W}$ 의 극한에서 정확한 동치가 성립하도록 하는 함수 $h(\mathbf{W}^*)$ 가 존재한다:

$$P(\mathbf{W}^*) = P(\mathbf{W}) + \frac{\partial P}{\partial \mathbf{W}}(\mathbf{W}^* - \mathbf{W}) + h(\mathbf{W}^*) \quad (31)$$

i^{th} 행의 가중치 변화가 $\Delta \mathbf{W}_i = \frac{\partial P}{\partial \mathbf{W}_i}$ 일 때의 보상을 $P(\mathbf{W}^*)$ 로, 그리고 임의의 다른 선택을 $P(\mathbf{W}^*)$ 로 두자. $\lambda \rightarrow 0$ 이므로 $\mathbf{W}^* \rightarrow \mathbf{W}$ 이며, 후회의 정의에 따라 다음과 같이 쓸 수 있다:

$$rgt_i = E_{\Delta \mathbf{W}^*} \left[\max_j \Delta \mathbf{W}^* \cdot i^* \left[\frac{\partial P}{\partial \mathbf{W}}(\mathbf{W}^* - \mathbf{W}) - \frac{\partial P}{\partial \mathbf{W}}(\mathbf{W}^* - \mathbf{W}) \right] \right] \quad (32)$$

이는 선택을 $\mathbf{W}^*$ 와 $\mathbf{W}_*$ 로 둔 식 (31)을 서로 빼서 얻어진다. 다음으로 $\mathbf{W}^* - \mathbf{W} = -\lambda \Delta \mathbf{W}$ 를 대입하고, $\frac{\partial P}{\partial \mathbf{W}} \Delta \mathbf{W} = [\frac{\partial P}{\partial \mathbf{W}_0} * \Delta \mathbf{W}_0, \dots, \frac{\partial P}{\partial \mathbf{W}_n} * \Delta \mathbf{W}_n]$ 로 전개하면 다음이 된다:

$$\frac{\partial P}{\partial \mathbf{W}}(\mathbf{W}^* - \mathbf{W}) - \frac{\partial P}{\partial \mathbf{W}}(\mathbf{W}_* - \mathbf{W}) = \lambda \left(\frac{\partial P}{\partial \mathbf{W}_i} * \Delta \mathbf{W}_i^* - \frac{\partial P}{\partial \mathbf{W}_i} * \Delta \mathbf{W}_i \right) + \lambda \sum_{j \neq i} \frac{\partial P}{\partial \mathbf{W}_j} * \Delta \mathbf{W}_j + \frac{\partial P}{\partial \mathbf{W}_i} * \Delta \mathbf{W}_i$$

상수 λ 는 제거할 수 있으며, 두 번째 항은 다른 행들의 가중치 $\mathbf{W}^* \cdot j \neq i$ 에만 의존한다. 이들은 독립이며 균등하게 분포한다고 가정하므로 기대값 $E * \Delta \mathbf{W}_j$ 하에서 제거할 수 있다. 따라서 다음을 얻는다:

$$rgt_i = E_{\Delta \mathbf{W}^*} \left[\max_j \Delta \mathbf{W}_i^* \left[\frac{\partial P}{\partial \mathbf{W}_i} * \Delta \mathbf{W}_i^* - \frac{\partial P}{\partial \mathbf{W}_i} * \Delta \mathbf{W}_i \right] \right] \quad (34)$$

마지막으로, 벡터 a , b 와 그 사이의 각 θ 에 대해, 내적의 크기는 $|a||b| \cos \theta$ 이며, 벡터가 서로 평행할 때(즉, $\theta = 0$ 이고 $\cos(\theta) = 1$) 최대가 됨을 사용한다. 본 경우 $\Delta \mathbf{W}_i = \kappa * \frac{\partial P}{\partial \mathbf{W}_i}$ (어떤 상수 $\kappa > 0$ 에 대해)일 때 최대가 된다. 따라서 $P(\Delta \mathbf{W}^*)$ 는 $\Delta \mathbf{W}_i^* = \kappa * \frac{\partial P}{\partial \mathbf{W}_i}$ 일 때 최대화된다. 최대점에서 $P(\Delta \mathbf{W}^*) = P(\Delta \mathbf{W}^*)$ 이므로, 명제가 성립한다.

12.7 손실 수렴 시뮬레이션

아래는 분리(disjoint)된 경쟁 서브그래프가 다수 그래프에 대하여 스테이크 수렴(stake-convergence)하는 과정을 시뮬레이션하는 Python 코드입니다.

```
import matplotlib.pyplot as plt
from scipy.sparse import random
from sklearn.preprocessing import normalize
import numpy as np
np.set_printoptions(precision=3, suppress=True)

def random_weight_matrix(n, density):
    rows = []
    for _ in range(n):
        row_i = random(1, n, density=density).A # Each row has density A
        row_i = normalize(row_i, axis=1, norm='l1') # Each row sums to 1
        rows.append(row_i)
        assert np.shape(row_i)[1] == n
        assert np.isclose(np.sum(row_i), 1.0, 0.001)
    W = np.concatenate(rows)
    assert np.shape(W)[0] == n and np.shape(W)[1] == n
    return W

def random_stake_vector(n):
    S = np.random.uniform(0, 1, size=(n, 1))
    S = (S / S.sum())
    S = np.reshape(S, [1, n])
    S = np.transpose(S)
    return S

def sigmoid(x):
    return 1 / (1 + np.exp(-x))

def get_gradient(n, W, R, S, A, C):
    gradient = []

    for i in range(n): grad_i = 0.0 for k in range(n): inner = 0.0 for j in range(n): inner += A[j] *
    S[j] * C[j, k] grad_i += C[i, k] * R[k] * np.exp(inner) * sigmoid(inner) * sigmoid(inner) + W[i, k] *
    (sigmoid(inner) - gradient[i]) gradient.append(grad_i) return np.concatenate(gradient)

def cabal_decay(
    nA: int = 5,
```

```

nB: int = 100,
n_blocks: int = 2000,
tau: float = 0.01,
learning_rate: float = 0.05
):
    """
    Measures the proportion of stake held by a disjoint sub-graph 'cabal' as a function of steps.

    Args:
        nA (int):
            Number of peers in the honest graph
        nB (int):
            Number of peers in the disjoint graph
        n_blocks (int):
            Number of blocks to run.
        temperature (float):
            temperature of the sigmoid activation function.
        tau(float):
            stake inflation rate.
        learning_rate (float):
            scaling term learning rate.

    Returns:
        history (list(float)):
            Dishonest graph proportion.
    """
    n = nA + nB

    # Randomized Matrix of weights. We create the subgraphs by concatenating two disjoint
    # and random square matrices.
    W = np.concatenate((
        np.concatenate((random_weight_matrix(nA, 0.9), np.zeros((nA, nB))), axis=1),
        np.concatenate((np.zeros((nB, nA)), random_weight_matrix(nB, 0.9)), axis=1)
    ), axis=0); print('W \n', W)

    # Randomized Vector of stake.
    # Subgraph A gets 0.51 distributed across values 1->n_a.
    # Subgraph B gets 0.49 distributed across values 1->n_b
    S = np.concatenate((
        random_stake_vector(nA)*0.51,
        random_stake_vector(nB)*0.49
    ), axis=0); print('S \n', np.transpose(S))

    # Scaling vector. A is multiplied by S before each iteration
    # to attain our scaled state.
    A = np.ones(n); print('A \n', A)

    # Matrix of connectivity. We use the true absorbing markov chain calculation
    # In practice this is too difficult to compute and opt for a depth d cut off.
    Wdiag = np.zeros((n, n))
    Wdiag[np.diag_indices_from(W)] = np.diag(W)
    Q = W - Wdiag

    C = np.maximum(np.linalg.pinv(np.identity(n) - Q), np.zeros((n,n)))
    C = np.where(C > 0, 1, -1); print('C \n', C)

    # Iterate through blocks
    history = []
    for block in range(n_blocks):

        # Scale the stake against our vector A.
        S_scaled = S * np.reshape(A, (n,1)); # print('S * A \n', S_scaled)

        # Compute the ranks W^T S.
        R = np.matmul(np.transpose(W), S_scaled) / np.sum(S_scaled) #; print('R \n', R)

        # Compute our trust scores sigmoid((C^T S) * temperature)
        T = 1 / (1 + np.exp(-np.matmul(np.transpose(C), S_scaled))) #; print('T \n', T)

        # Compute our chain loss.
        loss = -np.dot(np.transpose(R), (T - 0.5)) #; print('loss \n', loss[0][0])

```

```

# Our loss is negative, so we emit more stake.
if loss < 0.0:

    # Compute our incentive function.
    I = R * T

    # Distribute more stake.
    S = S + tau * (I / np.sum(I))

    # Measure the size of our cabal.
    S_Honest = np.sum(S[:nA])
    S_Cabal = np.sum(S[nB:])
    ratio = S_Cabal / (S_Honest + S_Cabal)
    history.append(ratio)

# Our loss is positive, so we update our scaling terms.
else:
    # Compute the gradient of our scaling terms.
    grad = get_gradient(n, W, R, S, A, C) #; print('Grad \n', grad)

    # Update our scaling terms, being sure not to have negative values.
    A = np.minimum(np.maximum(A + learning_rate * grad, np.zeros_like(A)), 1)

plt.plot(history)

cabal_decay()

```

비트텐서 Docs

비트텐서는 참가자들이 최고 수준의 디지털 자원을 생산하는 오픈 소스 플랫폼으로, 여기에는 컴퓨팅 파워, 저장 공간, 인공지능(AI) 추론 및 학습, AI의 생물학적 단백질 구조 예측, 금융 시장 예측 등 다양한 형태가 포함됩니다.

비트텐서는 개별적인 서브넷(subnets)으로 구성되어 있습니다. 각 서브넷은 독립적인 채굴자(miners) 커뮤니티(자산을 생산하는 역할)와 검증자(validators) 커뮤니티(채굴자의 작업을 평가하는 역할)로 이루어져 있습니다.

비트텐서 네트워크는 지속적으로 유동성을 발행하며, 이는 네트워크의 토큰인 TAO (τ) 형태로 참가자에게 분배됩니다. 분배 비율은 각 참가자의 기여 가치에 비례합니다. 참가자는 다음과 같습니다: 채굴자, 검증자, 서브넷 생성자, 스테이커

비트텐서 소개

비트텐서 플랫폼의 구성 요소

비트텐서 플랫폼은 다음과 같은 구성 요소로 이루어져 있습니다:

1. 서브넷 풀(Subnets Pool)

각 서브넷은 인센티브 기반의 경쟁형 마켓플레이스로, 인공지능과 관련된 특정 종류의 디지털 자원을 생산합니다. 각 서브넷은 자산을 생산하는 **채굴자(miners)** 커뮤니티와, 채굴자의 작업을 측정하여 품질을 보장하는 **검증자(validators)** 커뮤니티로 구성됩니다. 품질 평가는 서브넷 고유의 기준에 따라 이루어집니다.

각 서브넷은 독자적인 목표와 기준을 가진 자체 커뮤니티입니다. 예를 들어, OpenTensor 재단이 개발한 [텍스트 프롬프팅 서브넷](#)은 서브넷 내 검증자가 보낸 프롬프트에 대해 최적의 응답(prompt completion)을 생성하는 채굴자에게 인센티브를 제공합니다.

현재 존재하는 서브넷을 탐색하려면 [TAO.app의 서브넷 목록](#)을 확인하거나, [Discord의 비트텐서 서브넷 커뮤니티](#)에 참여해보세요.

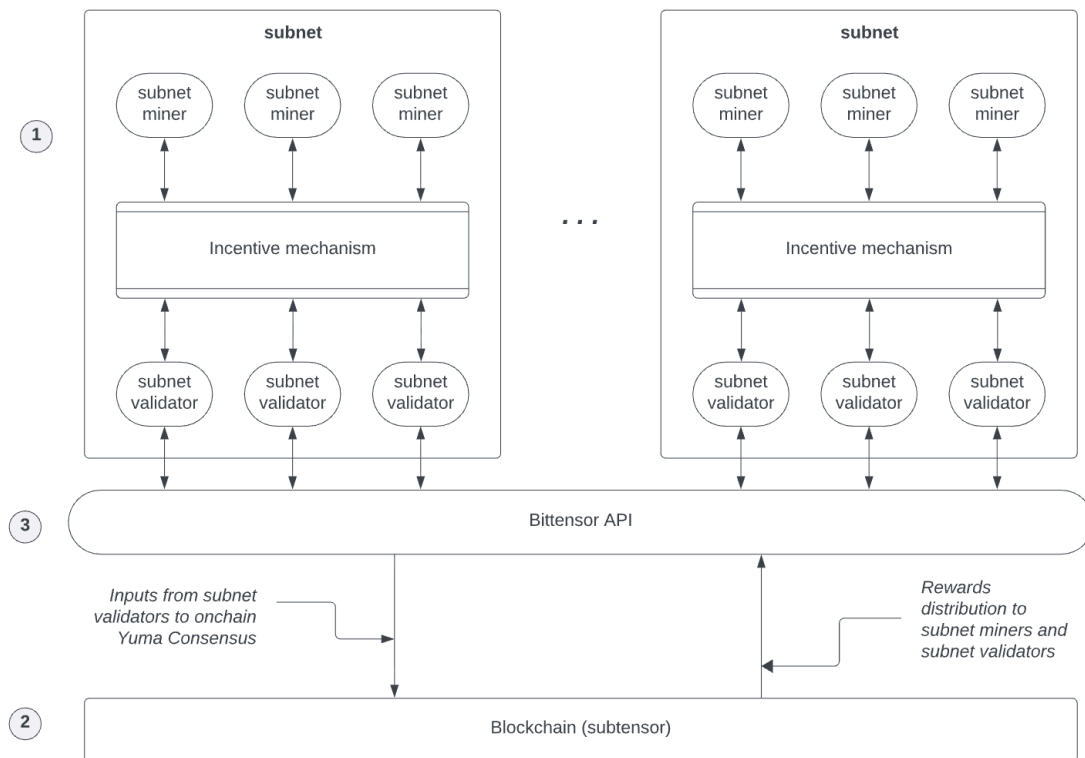
2. 비트텐서 블록체인

비트텐서 블록체인은 기록 시스템으로서의 역할을 하며, 그 토큰인 TAO (τ)는 서브넷 활동에 대한 인센티브로 사용됩니다. 채굴자와 검증자는 서브넷 내에서의 성과뿐 아니라, 해당 서브넷이 비트텐서 전체에서 달성한 성과에 따라 TAO를 획득합니다. 따라서 TAO의 발행은 채굴자와 검증자가 최상의 성과를 내도록 동기를 부여하며, 이를 통해 지속적인 개선이 가능한 이상적인 환경을 조성합니다. 비트텐서 블록체인은 채굴자, 검증자, 그리고 서브넷 생성자의 잔고 및 거래를 기록하며, 누구나 서브넷에 통화를 스테이킹하여 그들의 활동을 지원할 수 있도록 허용합니다.

3. 비트텐서 SDK

비트텐서 SDK는 서브넷 내에서 채굴자와 검증자 간의 상호작용을 지원하며, 모든 참가자가 블록체인과 필요한 방식으로 상호작용할 수 있도록 합니다.

비트텐서는 생태계 참여를 가능하게 하기 위해 모든 오픈 소스 도구, 코드베이스, 문서, 그리고 단계별 튜토리얼과 가이드를 제공합니다.



비트텐서 페르소나

다음의 역할들은 비트텐서에 참여할 수 있는 다양한 방식들을 정의하며, 본 웹사이트의 독스에서 각각의 역할이 다루어집니다:

- **채굴자(Miners)** — 디지털 자원을 생산하는 역할을 담당합니다. [비트텐서의 채굴](#)을 참조하세요.
- **검증자(Validators)** — 채굴자의 작업 품질을 평가합니다. [비트텐서의 검증](#)을 참조하세요.
- **서브넷 생성자(Subnet creators)** — 채굴자와 검증자가 수행하고 평가해야 할 작업의 인센티브 메커니즘을 관리합니다. [서브넷 생성하기](#)를 참조하세요.
- **스테이커(Stakers)** — TAO 보유자는 특정 검증자에게 TAO를 스테이킹하여 그들을 지원할 수 있습니다. [스테이킹 및 위임\(Staking and Delegation\)](#)을 참조하세요.

또한, 독스에서는 블록체인 운영자(Blockchain operator)의 역할도 설명합니다. 이 역할은 오프라인 테스트 시에만 해당되며, 사용자가 Subtensor를 비트텐서 메인넷(Main network)이나 테스트넷(Test network)에 연결되지 않은 독립적인 로컬 체인으로 실행할 때 적용됩니다.

[로컬에서 서브넷 실행하기\(Running a Subnet Locally\)](#)를 참조하세요.

💡 서브넷 탐색하기

[TAO.app](#)의 서브넷 목록에서 서브넷을 둘러보고, 각 서브넷의 코드 저장소 링크를 확인해보세요.

서브넷 개발

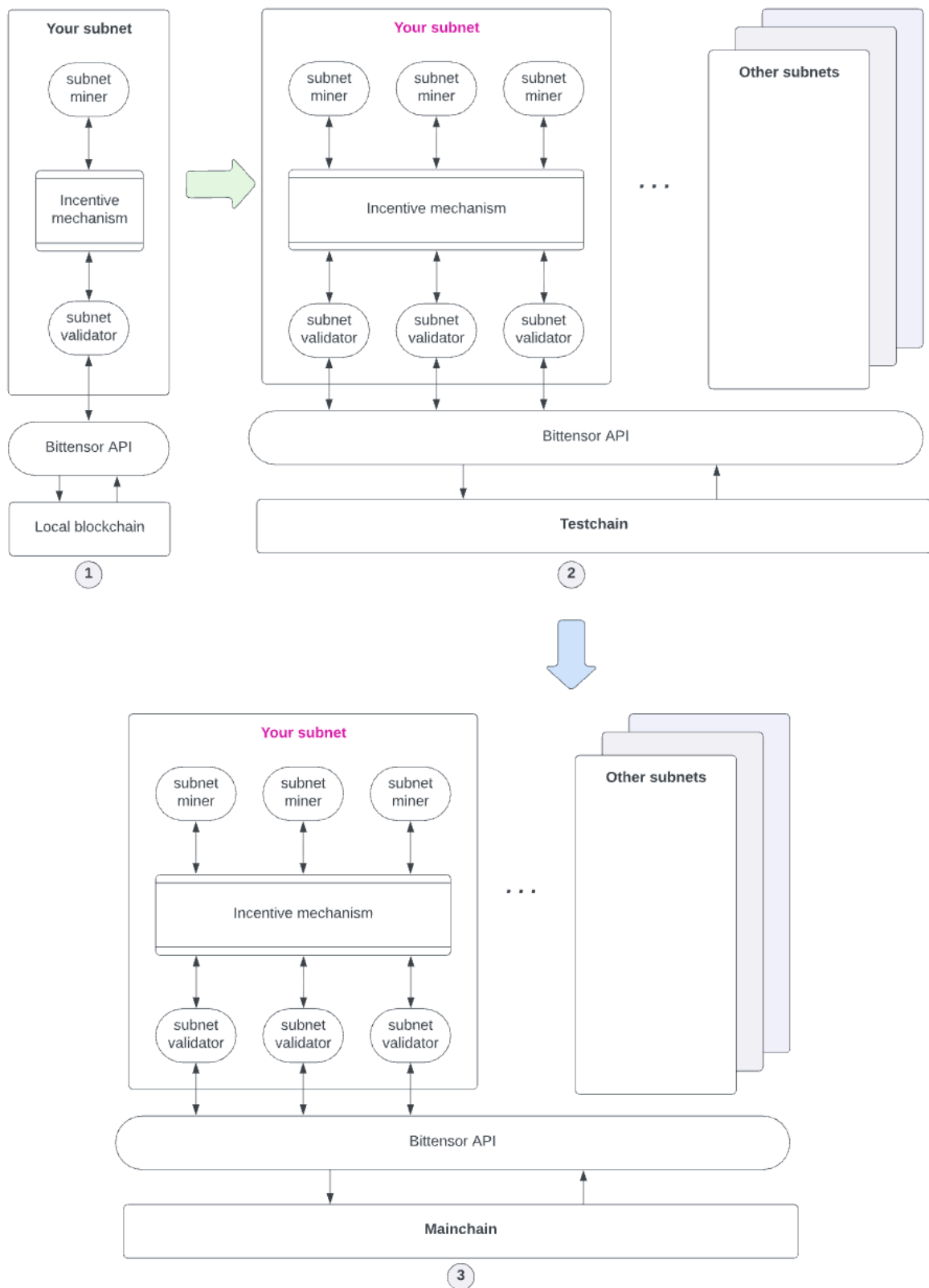
새로운 서브넷을 생성하든, 기존 서브넷에 참여하든, 항상 먼저 서브넷의 인센티브 메커니즘을 로컬 환경(Local)에서 테스트하고, 그 다음 비트텐서 테스트체인에서 테스트한 뒤, 마지막으로 비트텐서 메인체인에 연결하여 실제로 가동해야 합니다.

아래 개념적 배포 다이어그램은 이 세 단계를 시각적으로 보여줍니다.

인센티브 메커니즘을 테스트한다는 것은, 하나 이상의 검증자와 채굴자를 실행하여 토큰 발행(emission)이 의도된 대로 이루어지는지를 확인하는 것을 의미합니다.

서브넷에 참여하기 전

모든 서브넷 참가자는 서브넷에 참여하거나 새로운 서브넷을 생성하기 전에, 반드시 인센티브 메커니즘의 구조를 충분히 이해하는 것을 권장합니다.



FAQ

일반

비트텐서는 블록체인인가요, 아니면 AI 플랫폼인가요?

둘 다입니다!

비트텐서는 인공지능 추론(AI inference), 학습(training), 인프라(infrastructure)뿐만 아니라, 인공지능과 직접 관련되지 않은 다양한 디지털 자원을 생산하는 플랫폼입니다.

비트텐서는 자체 Substrate 기반 블록체인인 Subtensor 위에서 구동됩니다. 비트텐서 메인 네트워크의 분산 원장은 기록 시스템의 역할을 수행하며, 비트텐서의 암호화폐 토큰인 TAO는 플랫폼 전반의 활동에 대한 인센티브로 사용됩니다.

그렇다면 서브넷이란 무엇인가요?

서브넷은 디지털 자원을 경쟁적인 시장 구조 안에서 생산하는 커뮤니티이며, 비트텐서는 이러한 생산 과정에서 필요한 활동을 추적하고 인센티브를 제공합니다.

충분한 자금과 기술적 역량을 갖춘 누구나 새로운 서브넷을 생성하거나 기존 서브넷에 참여할 수 있습니다.

서브넷 내 경쟁은 어떻게 이루어지나요?

채굴자가 수행해야 할 작업은 서브넷 생성자가 정의한 인센티브 메커니즘에 의해 결정됩니다. 채굴자들은 해당 작업을 가장 잘 수행하기 위해 경쟁하며, 자신의 결과물을 검증자에게 제출합니다.

검증자는 서브넷 내에서 채굴자의 작업 품질을 평가하고 순위를 매깁니다. 이 평가 점수의 집계 결과에 따라 각 채굴자에게 발행되는 TAO의 양이 결정됩니다.

동시에 검증자 또한 자신의 평가 점수가 다른 검증자들의 일반적인 합의(consensus)와 얼마나 일치하는지에 따라 보상을 받기 때문에, 최선의 평가를 수행하도록 인센티브가 부여됩니다.

서브넷 채굴자의 작업은 구체적으로 무엇인가요?

채굴자의 작업은 서브넷마다 다릅니다. 일부 서브넷은 특화된 인공지능 추론, 학습, 또는 예측과 같은 AI 서비스를 제공합니다. 다른 서브넷은 저장(storage)이나 컴퓨팅(compute)과 같은 인프라 서비스를 제공합니다.

[TAO.app](#)에서 각 서브넷의 토큰노믹스(tokenomic) 정보를 탐색하고, [Learnbittensor.org](#)의 서브넷 목록에서 해당 서브넷이 지원하는 프로젝트와 서비스를 자세히 살펴보세요.

그렇다면 블록체인은 어디에서 역할을 하나요?

블록체인은 서브넷의 모든 주요 활동을 원장(ledger)에 기록합니다. 또한, Yuma Consensus (YC)라 불리는 알고리즘을 지속적으로 실행합니다. YC는 서브넷 검증자들이 부여한 채굴자들의 순위를 입력으로 받아, 채굴자, 검증자, 스테이커, 그리고 서브넷 생성자에게 발행될 TAO의 양을 계산합니다.

서브넷들 간의 통신이 가능한가요?

비트텐서 6.8.0 버전에서는 SubnetsAPI 라는 새로운 단순화된 베이스 계층(Abstract Base Class)이 도입되었습니다. 이를 활용하면 애플리케이션에서 서브넷 간 통신(cross-subnet communication)을 구현할 수 있습니다. 그러나 일반적으로 SubnetsAPI를 사용하지 않는 경우, Subtensor 블록체인은 한 서브넷의 데이터를 다른 서브넷의 데이터와 섞이지 않으며, 서브넷 간 직접적인 통신은 이루어지지 않습니다.

채굴 및 검증 관련

비트텐서의 채굴과 검증은 비트코인과 같은가요?

아닙니다. 몇 가지 중요한 차이점이 있습니다. 비트코인에서는 채굴자(miners)가 작업 증명(Proof of Work) 합의 알고리즘에 따라 블록을 검증하고, 이 블록들이 블록체인에 추가되도록 합니다.

반면, 비트텐서에서 서브넷 내의 “채굴(mining)”은 블록을 추가하는 것과는 관련이 없습니다. 여기서 채굴은 디지털 자원(digital commodities)을 생성하는 행위에 해당합니다. 마찬가지로, 비트텐서에서의 “검증(validation)”도 블록을 검증하는 행위가 아닙니다. 대신, 검증자는 채굴자들이 수행한 작업의 품질을 평가하고 검증하는 역할을 담당합니다.

그렇다면 비트텐서에는 별도의 블록체인 검증이 존재하나요?

그렇습니다. 비트텐서의 블록체인 자체의 검증(Validation)은 Opentensor Foundation이 권한 증명(Proof of Authority, PoA) 모델을 기반으로 수행합니다.

채굴자나 검증자, 혹은 서브넷을 생성할 유인은 무엇인가요?

비트텐서는 TAO 보상(emission)을 통해 네트워크 참여를 유도합니다. 매일 7,200 TAO가 네트워크에 발행되며, 약 12초마다 1 TAO씩 생성됩니다.

각 서브넷 내에서 TAO의 발행 비율은 다음과 같습니다:

- 서브넷 생성자(subnet creator): **18%**
- 검증자(validators): **41%**
- 채굴자(miners): **41%**

서브넷을 직접 만들고 싶지 않는데, 채굴자나 검증자만 될 수 있나요?

네, 가능합니다. 대부분의 참여자는 자신의 서브넷을 만들지 않습니다. 비트텐서에는 이미 참여할 수 있는 다양한 서브넷들이 존재합니다.

모든 서브넷의 채굴 및 검증 연산 요구사항을 한 곳에서 볼 수 있나요?

안타깝지만 그렇지 않습니다. 서브넷들은 Opentensor Foundation이 직접 운영하거나 관리하지 않으며, 각 서브넷의 환경은 지속적으로 변화합니다.

서브넷의 채굴자나 검증자로 영구적으로 남을 수 있나요?

계속 시도할 수는 있지만, 성과(performance)에 따라 결과가 달라집니다. 서브넷 내에서의 채굴 및 검증은 경쟁적입니다. 만약 채굴자나 검증자가 해당 서브넷 내에서 가장 낮은 순위의 세 명 중 하나로 평가되면, 그들은 해당 템포(tempo)가 종료될 때 등록 해제(deregistration)될 수 있으며, 다시 등록을 해야 합니다.

서브넷(Subnets) 이해하기

비트텐서에서 서브넷은 인공지능과 관련된 특정 종류의 디지털 자원을 생산하는 인센티브 기반 경쟁형 마켓플레이스입니다.

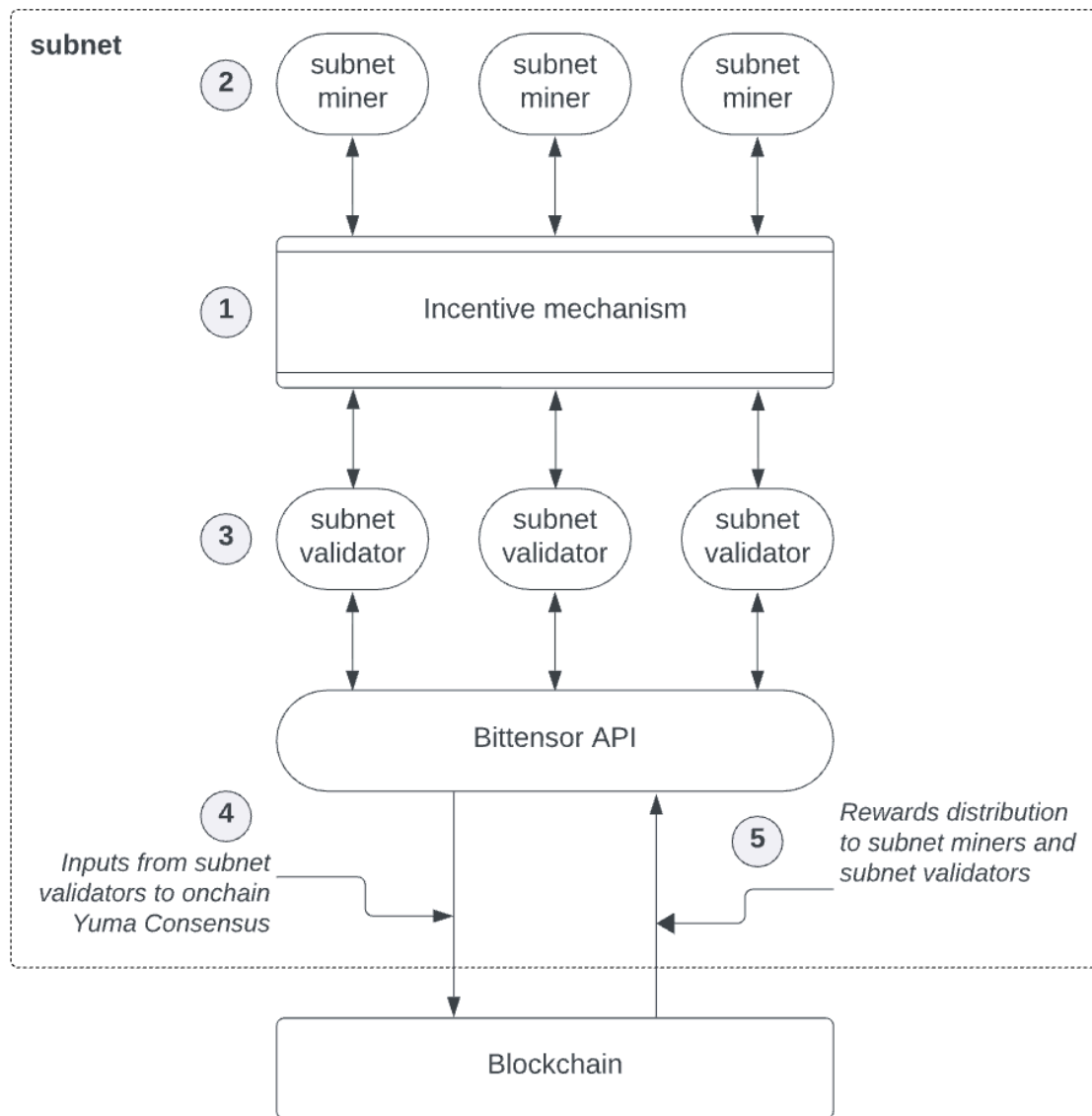
이는 자산을 생산하는 채굴자(miners) 커뮤니티와, 채굴자의 작업 품질을 평가하는 검증자(validators) 커뮤니티로 구성됩니다. 종종 검증자로 참여하는 주체들은, 채굴자가 제공하는 서비스를 활용하는 애플리케이션을 운영하기 위해 이 역할을 수행하기도 합니다.

비트텐서에서 발행되는 TAO (τ)는 서브넷 내에서의 성과뿐만 아니라, 비트텐서 전체 내에서의 서브넷 간 상대적 성과에 따라 채굴자와 검증자에게 분배됩니다.

서브넷의 구조

아래 그림은 서브넷의 주요 구성 요소를 보여줍니다:

1. 서브넷의 인센티브 메커니즘([Incentive Mechanism](#))은 채굴자가 수행해야 할 작업과, 검증자가 채굴자의 작업을 평가하기 위해 수행해야 할 작업을 정의합니다. 인센티브 메커니즘은 서브넷 고유의 요소이며, 서브넷 생성자에 의해 오프체인(off-chain) 코드 저장소 형태로 유지됩니다. 이 저장소는 채굴자와 검증자가 참여하기 위한 인터페이스를 정의합니다. 예시로 [Subnet 1](#)을 참조하세요.
2. **채굴자(Miners)**는 인센티브 메커니즘에 정의된 대로 유용한 작업을 수행합니다. 예를 들어 Subnet 1에서는 채굴자가 채팅 프롬프트 응답(chat prompt completion)을 제공합니다.
3. **검증자(Validators)**는 서브넷의 인센티브 메커니즘에 정의된 기준에 따라 채굴자가 수행한 작업을 독립적으로 평가합니다.
4. 각 검증자는 최근 기간 동안 각 채굴자의 성과를 점수화합니다. 각 검증자가 각 채굴자에게 부여한 점수 행렬(matrix)은 **Yuma Consensus** 알고리즘의 입력으로 사용됩니다.
5. **Yuma Consensus** 알고리즘은 온체인(on-chain)에서 작동하며, 성과를 기반으로 채굴자, 검증자, 그리고 서브넷 생성자에게 발행되는 토큰을 결정합니다.



유동성 풀

각 서브넷은 자체적인 자동화된 마켓 메이커(Automated Market Maker, AMM)로 작동하며, 두 가지 유동성 준비금을 가집니다. 하나는 비트텐서 네트워크의 통화인 **TAO (τ)**를 포함하고, 다른 하나는 서브넷 고유의 “동적(dynamic)” 통화인 **알파(α)** 토큰을 포함합니다. 알파 토큰은 TAO를 서브넷의 준비금(reserve)에 스테이킹(staking)함으로써 구매할 수 있으며, 해당 준비금은 매우 작은 초기 유동성(1×10^{-9})으로 시작합니다.

따라서 서브넷의 경제는 다음의 세 가지 통화 풀로 구성됩니다:

- **TAO 준비금(Tao Reserves)** — 서브넷에 스테이킹된 TAO (τ)의 양.
- **알파 준비금(Alpha Reserves)** — 구매 가능한 알파(α)의 양.
- **유통 중인 알파(Alpha Outstanding)** — 서브넷 참가자의 핫키(hotkey)에 보관된 알파(α)의 양으로, 서브넷 내 총 스테이크(total stake)라고도 합니다.

용어: 알파 토큰(Alpha Tokens)

각 서브넷은 고유한 이름을 가진 자체 통화를 가지지만, 일반적으로 특정 서브넷의 토큰은 그 서브넷의 **α 토큰**이라고 부릅니다. 예를 들어 서브넷 α 의 토큰을 α , 서브넷 β 의 토큰을 β , 서브넷 γ 의 토큰을 γ 로 표기합니다.

이러한 서브넷 토큰은 비트텐서 전체 네트워크의 토큰인 TAO (τ)와 구별됩니다. 서브넷 풀의 준비금 비율(TAO/alpha)은 해당 서브넷의 알파 토큰 가격을 결정합니다.

서브넷의 **알파 토큰 가격**은 해당 서브넷의 TAO 준비금 대비 알파 준비금의 비율에 의해 결정됩니다. 서브넷 참가자의 핫키에 보관되어 있지만 준비금에 포함되지 않은 알파 통화는 **유통 중인 알파(Alpha Outstanding)**로 불립니다.

Dynamic TAO 기능이 활성화된 `btcli`에서 `btcli subnet list` 명령어를 실행하면, 비트텐서 테스트넷에서 각 서브넷의 통화 준비금 및 관련 정보를 확인할 수 있습니다.

Netuid	Name	Price (τ_{in}/α_{in})	Market Cap ($\alpha \times \text{Price}$)	Emission (τ)	P (τ_{in}, α_{in})	Stake (α_{out})	Supply (α)	Tempo (k/n)
0	τ root	1.00 τ/T	τ 5.93m	τ 0.0000	-, -	T 5.93m	5.93m T / 21M	- / -
3	γ templar	0.02 τ/γ	τ 57.32	τ 0.0197	τ 31.44, 1.43k γ	1.18k γ	2.61k γ / 21M	67 / 99
9	ι pretrain	0.02 τ/ι	τ 55.38	τ 0.0194	τ 30.91, 1.46k ι	1.16k ι	2.61k ι / 21M	73 / 99
1	α apex	0.02 τ/α	τ 54.45	τ 0.0192	τ 30.65, 1.47k α	1.14k α	2.61k α / 21M	65 / 99
2	β omron	0.02 τ/β	τ 54.45	τ 0.0192	τ 30.65, 1.47k β	1.14k β	2.61k β / 21M	66 / 99
4	δ targon	0.02 τ/δ	τ 54.45	τ 0.0192	τ 30.65, 1.47k δ	1.14k δ	2.61k δ / 21M	68 / 99

알파 토큰의 가격/비율

이상적인 가격

각 서브넷의 **가격(Price)**은 준비금 내 TAO (τ_{in}) 대비 알파 (α_{in})의 함수로 정의됩니다:

$$Price = \frac{T_{in}}{\alpha_{in}}$$

비트텐서 내 발행

비트텐서의 토큰 경제에서는 성장 촉진과 가격 안정화, 그리고 시장 조작 방지를 위해 설계된 알고리즘에 따라 유동성이 지속적으로 발행(emitted)됩니다.

각 블록마다 다음이 수행됩니다:

- 체인은 TAO를 발행하여 각 서브넷의 TAO 준비금에 투입합니다.
- 체인은 기본 알파 발행 속도의 두 배로 알파 토큰을 발행합니다(기본 속도는 블록당 1 α 이며, TAO와 동일한 반감기 스케줄을 따릅니다). 이때 발행된 알파 토큰은 다음 두 부분으로 할당됩니다:
 - 서브넷의 알파 준비금(유동성 증가)
 - 유통 중인 알파(채굴자, 검증자, 서브넷 생성자에 대한 인센티브)

서브넷의 분산 평가

비트텐서 내에서 서브넷의 상대적 가치, 즉 가중치(weight)는 매우 중요합니다. 이는 서브넷 간 발행량과 그 서브넷 내 채굴자 및 검증자에게 분배되는 양을 결정하기 때문입니다. Dynamic TAO 도입 이전에는, 비트텐서 네트워크 내 서브넷 간의 상대적 가중치는 Root Network 검증자들의 평가에 대한 Yuma Consensus 결과로 결정되었습니다. 이로 인해 루트 네트워크 검증자 키를 보유한 주체들이 본질적으로 중앙집중적인 역할을 수행하게 되었습니다.

그러나 Dynamic TAO에서는, 각 서브넷의 안정화된 토큰 가격으로 표현되는 시장 가치의 자연스러운 형성에 따라 서브넷의 상대적 가중치가 결정됩니다. TAO 보유자는 TAO를 서브넷에 스테이킹하여 해당 서브넷의 동적 통화인 알파(α) 토큰을 얻을 수 있습니다. 이 방식에서 스테이커는 자신의 TAO로 “투표”를 수행함으로써 서브넷의 가치를 결정하고, 그 결과로 해당 서브넷의 검증자와 채굴자에게 배분되는 발행량이 결정됩니다. 스테이커는 그 대가로 해당 서브넷의 발행분 일부를 보상으로 받습니다.

서브넷 제로(Subnet Zero)

Dynamic TAO에서 서브넷 제로(Subnet Zero) — 또는 *루트 서브넷(Root Subnet)* — 은 특별한 서브넷입니다. 이 서브넷은 자체적인 α 화폐를 가지지 않는 유일한 서브넷입니다. 채굴자(miner)는 서브넷 제로에 등록할 수 없으며, 검증(validation) 작업도 수행되지 않습니다. 그러나 검증자(validators)는 등록할 수 있으며, 다른 서브넷과 마찬가지로 τ 보유자(τ -holders)가 해당 검증자에게 스테이킹(staking)을 할 수 있습니다. 이를 통해 τ 보유자는 특정 서브넷에 종속되지 않은 상태(subnet-agnostic)로 검증자에게 τ 를 스테이킹할 수 있는 메커니즘이 제공됩니다. 이 구조는, 검증자의 서브넷 내 가중치(weight)가 해당 서브넷의 α 보유 비율뿐 아니라 루트 서브넷에 스테이킹된 TAO의 비율도 포함하기 때문에 가능합니다.

시간이 지남에 따라, 서브넷 제로에 스테이킹된 TAO로부터 발생하는 보상(emission)은 활성 서브넷의 α 화폐 기반 스테이크에 비해 상대적으로 감소하게 됩니다.

검증자 스테이크 가중치

검증자의 서브넷 내 스테이크 가중치는 다음과 같이 계산됩니다. 즉, 검증자의 α 스테이크와 τ 스테이크에 tao_weight 파라미터를 곱한 값의 합입니다.

$$\text{Validator stake weight} = \text{Alpha stake}(\alpha) + \text{TAO stake}(\tau) \times \text{TAO weight}$$

참고: 서브넷 제로에서 검증자의 스테이크 가중치는 **그의 스테이킹된 TAO 값과 동일합니다.**

검증자의 상대적 스테이크 가중치(relative stake weight) — 즉, 서브넷 내 전체 스테이크 가중치 대비 자신의 비율 — 는 해당 검증자가 채굴자(miner)를 평가할 때 가지는 투표 권한(voting power)과 그 검증자가 받을 보상(emission)의 비율을 결정합니다.

$$\text{Validator } x \text{의 상대적 스테이크 가중치} = \frac{\alpha_x + \tau_x \times w_\tau}{\sum_{v \in \text{validators}} (\alpha_v + \tau_v \times w_\tau)}$$

인센티브 메커니즘

이 파트에서는 비트텐서에서의 인센티브 메커니즘(incentive mechanisms)의 개념과 사용 방식을 살펴봅니다.

각 서브넷은 하나 이상의 인센티브 메커니즘을 갖고 있으며, 이는 참가자의 행동과 서브넷의 디지털 자원 생산을 유도하는 평가 모델(scoring model)로서 작동합니다. 이 메커니즘은 검증자가 채굴자의 작업을 어떻게 평가해야 하는가를 정의합니다. 채굴자는 검증자로부터 높은 점수를 받기 위해 이러한 모델에 맞게 자신의 작업을 최적화하며, 이는 더 높은 발행량(emission)으로 이어집니다. 반면, 검증자는 모델에 따라 채굴자의 작업을 정확하게 평가하도록 인센티브를 받습니다. 왜냐하면 검증자가 다른 검증자들의 평가 합의(consensus)에서 벗어나면, 해당 불일치에 대해 낮은 발행량으로 페널티를 받기 때문입니다.

서브넷이 여러 인센티브 메커니즘을 사용하는 경우, 각 메커니즘은 독립적으로 작동하며 Yuma Consensus 계산을 위한 자체 본드 풀(bond pool)을 가집니다. 이를 통해 서브넷 생성자는 서로 다른 작업 유형이나 평가 기준에 따라 발행량을 분배할 수 있습니다.

서브넷의 각 검증자는 주기적으로 각 채굴자에게 부여할 가중치 벡터(weight vector)를 계산해야 합니다. 이 벡터는 각 인센티브 메커니즘 내에서 채굴자의 성과에 따른 종합 순위를 나타냅니다. 검증자는 이러한 가중치 벡터(weight vectors)를 블록체인으로 전송합니다.

비트텐서 블록체인은 특정 서브넷의 모든 검증자로부터 최신 순위(가중치 벡터)를 수신한 후, 이를 기반으로 각 인센티브 메커니즘별 가중치 행렬(weight matrices)을 형성합니다. 이 행렬은 온체인 상의 Yuma Consensus 모듈에 입력값으로 제공됩니다. Yuma Consensus(YC)는 이러한 가중치 행렬과, 서브넷 내 각 UID에 연관된 스테이크(stake) 양을 활용하여, 각 참가자에게 분배될 발행량을 계산합니다. 이 발행량은 각 템포(tempo) — 즉, 360 블록 — 의 종료 시점에 확정되어 참가자의 핫키(hotkey)에 반영됩니다.

참고: 템포의 길이(360 블록)는 모든 사용자 생성 서브넷에 동일합니다. 그러나 각 서브넷이 생성된 시점에 따라 템포의 실제 타이밍은 서로 다를 수 있습니다.

다중 인센티브 메커니즘

서브넷은 채굴자의 성과를 다양한 측면에서 평가하거나, 하나의 서브넷 내에서 여러 종류의 작업을 지원하기 위해 여러 인센티브 메커니즘을 구현할 수 있습니다. 이러한 구조를 통해 서브넷 생성자는 다음과 같은 이점을 얻습니다:

- **다양한 작업에 대한 발행량 분배:** 서브넷의 총 발행량 중 일부를 특정 작업 유형에 할당할 수 있습니다.
- **독립적인 평가 유지:** 각 메커니즘은 별도의 본드 풀에서 작동하므로, 한 메커니즘에서의 채굴자 성과가 다른 메커니즘의 평가에 영향을 미치지 않습니다.
- **전문화된 경쟁 촉진:** 채굴자는 자신이 잘할 수 있는 특정 메커니즘에서 두각을 나타낼 수 있습니다.
- **투명한 제어 가능성 제공:** 모든 발행 분배 및 메커니즘 구성은 온체인에서 투명하게 확인할 수 있습니다.

서브넷 생성자의 책임

서브넷이 성공적으로 운영되기 위해서는, 생성자가 다음의 역할을 수행해야 합니다:

- 채굴자가 점수(즉, 발행량)를 최적화하려는 본능을 바람직한 방향으로 유도할 수 있는 인센티브 메커니즘을 설계해야 합니다.
- 채굴자 및 검증자의 참여를 촉진해야 합니다.
- 채굴자 또는 검증자에 의한 시스템 악용(exploit)을 방지해야 합니다.

참여 유도

우수한 성과를 내는 서브넷 채굴자와 검증자를 유치하기 위해, 서브넷에 대한 충분한 문서를 공개하는 것이 중요합니다.

서브넷 검증자가 귀하의 서브넷에 등록하면, 검증자 모듈을 실행하여 검증 작업을 시작합니다. 마찬가지로, 서브넷 채굴자 역시 등록 후 채굴자 모듈을 실행합니다. 예를 들어, 텍스트 프롬프팅 서브넷(text prompting subnet)의 다음 문서를 참조하면 이러한 절차를 빠르게 이해할 수 있습니다:

- [검증자 실행하기\(Running a validator\)](#)
- [채굴자 실행하기\(Running a miner\)](#)

악용(Exploits) 방지

잘 설계된 인센티브 메커니즘은 경쟁이 채굴자의 지속적인 개선을 이끄는 선순환을 촉진합니다. 반대로, 결함 있는 메커니즘은 지름길과 악용을 부추겨 서브넷의 무결성을 저해하고 참여를 약화시킵니다.

인센티브 메커니즘이 게임화(gaming)되거나 의도하지 않은 숏컷이 발생하지 않도록, 정교한 모델링과 테스트, 커뮤니티 검증을 통해 면밀히 점검해야 합니다. 서브넷 생성자는 채굴자 성과를 지속적으로 모니터링하고, 필요에 따라 메커니즘을 업데이트하여 미세 조정할 것을 예상해야 합니다.

인센티브 메커니즘 설계

다른 서브넷 생성자들이 무엇을 하고 있는지 조사해야 합니다. TAO.app의 서브넷 목록](#taoapps-subnets-listings)을 둘러보며 다른 서브넷과 그 코드 저장소 링크를 탐색하고, 서브넷 설계에 관한 최신 연구(활발히 진행 중인 분야)를 학습해야 합니다.

인센티브 메커니즘의 구성 요소

서브넷 인센티브 메커니즘은 다음을 제공해야 합니다:

- 검증자가 채굴자에게 쿼리를 보내는 방식과 채굴자가 응답하는 방식에 대한 프로토콜
- 채굴자가 수행해야 할 작업에 대한 태스크 정의
- 검증자가 채굴자의 작업을 평가하는 데 사용할 채점 메커니즘

채굴자-검증자 프로토콜

서브넷 생성자는 검증자가 채굴자에게 어떻게 쿼리해야 하는지, 그리고 채굴자가 어떻게 응답해야 하는지에 대한 프로토콜을 정의해야 합니다. 프로토콜은 Axon-Dendrite 클라이언트-서버 모델과 Synapse 데이터 오브젝트를 사용하여 구축됩니다.

서브넷 태스크

태스크는 채굴자가 수행할 작업을 정의하기 때문에, 모든 인센티브 메커니즘의 핵심 구성 요소 중 하나입니다. 태스크의 예로는 자연어 프롬프트에 대한 응답과 암호화된 파일 저장이 있습니다. 태스크는 서브넷이 생산하는 디지털 자원의 의도된 사용 사례를 포착하도록 설계되어야 합니다. 일반적으로 태스크는 서브넷과 상호작용하는 사용자의 실제 상호작용을 현실적으로 모사해야 합니다.

서브넷 평가 모델

태스크가 채굴자가 무엇을 해야 하는지를 설명한다면, 평가 모델은 어떻게 해야 하는지를 규정합니다. 마찬가지로, 태스크가 사용자 상호작용을 모사해야 하는 것처럼, 평가 모델은 사용자 선호나 원하는 결과를 모사해야 합니다.

다른 모든 머신러닝 모델과 마찬가지로, 서브넷에는 지속적으로 최적화되는 목적 함수가 존재합니다. 평가 모델은 서브넷 내 모든 채굴자 행동(의도된 것과 의도되지 않은 것 모두)의 품질을 정의합니다.

운영 관점에서, 평가 모델은 채굴자의 응답을 수치 점수로 변환하는 수학적 객체입니다. 평가 모델은 의도된 태스크와 채굴자의 행동을 정렬시키는 데 필요한 만큼 많은 다양한 평가 메커니즘을 포함할 수 있습니다.

채굴자는 자신의 발행량을 결정하기 때문에 가능한 한 가장 높은 점수를 달성하기 위해 지속적으로 경쟁할 것입니다. 점수에 상한이 설정되어 있다면, 채굴자는 더 이상 개선할 동기를 느끼지 못할 수 있습니다. 따라서 정체가 아니라 채굴자의 지속적인 개선이 가능하도록 주의 깊게 설계해야 합니다.

인센티브 메커니즘 설계 시 추가 고려사항

- 채굴자(miner)의 하드웨어를 고려해야 합니다. 이는 출력이나 성능 평가에서 차이를 초래할 수 있습니다.
- 정확한 재현성이 어려운 경우, 유사성 임계값(similarity threshold)을 엄격하게 설정해야 합니다. 예를 들어, [CLIP](#)과 같은 임베딩(embedding)을 비교하거나, 지각적 해싱(perceptual hashing)을 적용할 수 있습니다.
- 채굴자가 결과를 사전에 계산(precompute)하거나 캐싱(caching)하는 것을 방지하기 위한 조치를 취해야 합니다.
 - 검증자(validator)가 제공하는 무작위 시드(random seed)를 사용하고, 채굴자가 출력의 모든 무작위성을 제어하지 못하게 해야 합니다.
 - 작은 입력 변동(input variation)을 도입해야 합니다.
 - 평가 로직(scoring logic)이 부분적인 출력이 완전한 결과와 유사하게 평가되지 않도록 해야 합니다.
- 속도, 신뢰성, 품질 간의 균형을 신중하게 고려해야 합니다.
- 검증 과정에 자연적 질의(organic queries)를 포함하는 방안을 고려해야 합니다.
- 인센티브 메커니즘을 개선하거나 새로운 취약점(exploit) 및 비효율적 요소를 해결하기 위한 업데이트를 자주 배포해야 함을 예상해야 합니다. 최적의 서브넷(subnet)들은 새로운 하드웨어, 새로운 모델, 그리고 새로운 채굴자 행동을 반영하기 위해 빈번히 업데이트됩니다.
- 다른 서브넷을 관찰해야 합니다.
다중 모델 전략(multi-model strategy)이나 부분적 무작위화(partial randomization)와 같은 여러 솔루션을 자체 시스템에 맞게 적용할 수 있습니다.

서브넷 내의 다중 인센티브 메커니즘

기존에는 각 서브넷이 단일 인센티브 메커니즘(incentive mechanism)으로 운영되었습니다. 이는 검증자가 채굴자의 작업 가치를 기반으로 가중치(weight)를 부여하기 위해 실행하는 함수입니다. 이제 서브넷은 다중 인센티브 메커니즘을 지원하여, 서브넷 생성자가 여러 평가 기준에 따라 발행량을 분배할 수 있습니다. 각 메커니즘은 Yuma Consensus를 독립적으로 실행하며, 서로 다른 작업에 대한 채굴자의 성과를 평가하기 위해 별도의 결속 풀(bond pool)을 유지합니다.

각 채굴자는 인센티브 메커니즘별로 별도의 발행량을 받으므로, 한 메커니즘에서의 성과가 다른 메커니즘의 평점에 영향을 미치지 않습니다. 각 에포크(epoch)의 발행량은 모든 메커니즘의 결과를 합산하여 계산됩니다. 검증자는 모든 인센티브 메커니즘에서의 성과를 가중합(weighted sum) 형태로 배당(dividend)받습니다. 검증자가 일부 메커니즘만 검증할 경우, 그에 비례하여 발행량이 감소합니다. 다중 인센티브 메커니즘은 서브넷 전체의 총 발행량을 변경하지는 않지만, 서브넷 생성자가 서로 다른 작업을 수행하는 채굴자에게 그 발행량을 배분할 수 있는 구조를 제공합니다. 이 메커니즘은 서브넷 생성자가 인센티브를 제공하려는 작업에 대해 세밀하고 투명한 온체인(on-chain) 제어를 수행할 수 있게 하며, 채굴자의 노력을 시기별로 가장 필요한 작업에 집중시킬 수 있도록 합니다.

각 인센티브 메커니즘은 다음과 같은 고유 요소를 가집니다.

- **가중치 행렬(weight matrix):** 각 검증자는 서브넷의 각 인센티브 메커니즘에서 각 채굴자에게 가중치를 설정합니다.
- **독립적인 결속 풀:** 각 메커니즘은 Yuma Consensus 계산을 위해 별도의 결속 관계를 유지합니다.
- **독립적인 발행량:** 검증자가 설정한 가중치에 따라 각 채굴자의 메커니즘별 발행량은 독립적으로 결정됩니다.
- **투명한 온체인 데이터:** 모든 인센티브 메커니즘 구성과 발행량 흐름은 온체인에서 확인할 수 있습니다.
- **발행량 분배:** 서브넷 생성자는 `sudo_set_mechanism_emission_split` extrinsic을 사용하여 각 메커니즘에 전체 발행량 중 몇 퍼센트를 할당할지 제어할 수 있습니다.

핵심 요약

1. **동일한 검증자(Same Validators), 동일한 지분(Same Stake):** 모든 검증자는 서브넷 내의 모든 인센티브 메커니즘에 동일한 지분 가중치(stake weight)로 참여합니다.
2. **동일한 채굴자(Same Miners):** 서브넷에 등록된 모든 채굴자는 해당 서브넷의 어떤 인센티브 메커니즘에도 참여할 수 있습니다.
3. **소유자 제어 비율(Owner-Controlled Proportions):** 서브넷 생성자 키(subnet creator key) 보유자가 인센티브 메커니즘 간의 발행량 분배(emission distribution)를 설정합니다.
4. **분리된 결속 풀(Separate Bond Pools):** 각 인센티브 메커니즘은 독립적인 Yuma Consensus 계산을 위해 별도의 결속 관계(bonding relationship)를 유지합니다.

스테이커가 알아야 할 것

핵심 영향:

- **스테이킹 메커니즘에는 변화 없음:** 스테이킹된 지분은 서브넷 내의 검증자에게 위임(delegate)되며, 모든 인센티브 메커니즘에 동일하게 적용됩니다.
- **총 발행량은 동일함:** 서브넷의 총 발행량은 변하지 않으며, 다중 메커니즘은 단지 채굴자와 검증자 간 내부적으로 이 발행량을 재분배할 뿐입니다.
- **투명한 분배 구조:** 모든 발행 비율(emission proportion)은 온체인(on-chain)에서 확인 가능하므로, 서브넷 소유자가 발행량을 어떻게 분배하는지 명확히 볼 수 있습니다.

스테이킹 전략에 의미하는 바

- **즉각적인 조치 불필요:** 현재의 스테이킹 전략을 변경할 필요는 없습니다.
- **향상된 모니터링:** 서브넷의 건전성을 이해하기 위해 개별 메커니즘의 성과(performance)를 추적하는 것이 좋습니다.
- **위험 평가:** 서브넷의 품질을 평가할 때 인센티브 메커니즘의 설계 구조를 고려하십시오.
- **커뮤니티 감시:** 투명성을 활용하여 서브넷 소유자가 공정하게 발행량을 분배하고 있는지 감시할 수 있습니다.

채굴자가 알아야 할 것

자동 참여

- **별도의 등록 절차 없음:** 서브넷에 등록하면, 해당 서브넷의 모든 인센티브 메커니즘에 자동으로 참여할 수 있습니다.
- **모든 메커니즘에 동일한 UID 사용:** 서브넷 내의 모든 인센티브 메커니즘에서 동일한 UID를 사용합니다.

성과 추적

- **독립적인 평가:** 각 메커니즘에서의 성과 평가는 서로 독립적입니다. (예: 메커니즘 0의 평가가 메커니즘 1의 점수에 영향을 주지 않음)
- **분리된 인센티브 열(column):** 메타그래프(metagraph) 데이터에서 각 메커니즘별 개별 인센티브 금액을 확인할 수 있습니다.
- **누적 발행량:** 총 발행량은 참여 중인 모든 메커니즘의 발행량을 합산한 값입니다.

검증자가 알아야 할 것

핵심 변화

- **분리된 가중치 설정:** 각 인센티브 메커니즘별로 가중치를 독립적으로 설정해야 합니다.
- **독립적인 평가:** 각 메커니즘은 고유한 평가 기준에 따라 별도로 평가되어야 합니다.
- **분리된 결속 풀:** 각 메커니즘은 Yuma Consensus 계산을 위해 독립적인 결속 관계를 유지합니다.
- **동일한 스테이크 가중치:** 모든 메커니즘에서 동일한 스테이크 가중치가 적용되며, 추가 스테이크는 필요하지 않습니다.
- **가중 배당 계산:** 검증자의 배당(dividend)은 모든 메커니즘에서의 성과를 가중합한 값으로 계산됩니다. 모든 메커니즘을 검증하지 않는 경우, 이에 비례하여 발행량(emission)이 감소합니다.

운영상의 변경사항(Operational Changes)

1. 평가 작업

- **다중 평가:** 검증자는 각 메커니즘(mechanism)의 과제(task)별로 채굴자(miner)를 개별적으로 평가해야 합니다.
- **상이한 평가 기준:** 각 메커니즘은 서로 다른 평가 기준(evaluation standards)을 가질 수 있습니다.

2. 데이터 구조 변경(Data Structure Changes)

- **2차원 가중치:** 이제 가중치는 각 채굴자(miner) 및 각 메커니즘 단위로 설정됩니다.
- **분리된 인센티브 추적:** 각 메커니즘은 자체적으로 독립적인 인센티브 추적 시스템을 가집니다.
- **확장된 메타그래프:** 새로운 메커니즘별 가중치 및 인센티브 열(columns)이 추가되었습니다.

서브넷 생성자 및 개발자가 알아야 할 것

핵심 변화

- **보상 분배:** 총 보상(emission)의 몇 퍼센트를 각 인센티브 메커니즘에 할당할지를 `sudo_set_mechanism_emission_split` 명령을 통해 제어할 수 있습니다. 메커니즘의 개수가 새로 설정되면, 보상 분배 비율은 기본적으로 균등 분배(even split)로 초기화 됩니다. 이후 필요에 따라 사용자 정의(custom) 비율로 다시 설정할 수 있습니다.
- **인센티브 메커니즘 설계:** 각 메커니즘별로 수행해야 할 구체적인 작업(task)과 평가 기준(evaluation criteria)을 서브넷 소유자가 직접 정의 합니다.
- **투명한 설정:** 모든 메커니즘 설정값은 온체인(on-chain)에서 투명하게 공개되며, 커뮤니티가 이를 직접 검토하고 감시할 수 있습니다.
- **단일 서브넷 슬롯(Single Subnet Slot):** 여러 개의 경쟁(competition)이나 보상 구조를 운영하기 위해 여러 서브넷을 별도로 등록할 필요가 없습니다.
- **즉시 반영되는 메커니즘 개수 설정:** 메커니즘의 개수가 변경되면, 그 설정은 즉시 반영됩니다.
- **속도 제한:** 메인넷(Mainnet)에서는 서브넷 소유자가 7,200 블록(약 24시간)마다 한 번씩만 메커니즘 개수를 변경할 수 있습니다.

발행량 분배 예시

각 서브넷에서 서브넷 생성자는 발행량의 18%를 유지하고, 41%는 채굴자에게, 41%는 검증자 및 그들의 스테이커에게 할당됩니다. 단, 서브넷 생성자가 자신의 몫을 줄였을 경우 예외가 있을 수 있습니다. 이 중 채굴자와 검증자에게 분배되는 41%를 세 가지 인센티브 메커니즘에 따라 배분한 예시는 다음과 같습니다. (서브넷에서 100 τ 가 발생했다고 가정)

정보 (Info)

현재는 하나의 서브넷당 최대 2개의 메커니즘만 허용되지만, 향후 이 상한이 확대될 예정입니다.

- **메커니즘 0 (60%):** $100 \tau \times 0.41 \times 0.6 = 24.6 \tau$
- **메커니즘 1 (30%):** $100 \tau \times 0.41 \times 0.3 = 12.3 \tau$
- **메커니즘 2 (10%):** $100 \tau \times 0.41 \times 0.1 = 4.1 \tau$

또한, 특정 메커니즘(예: 메커니즘 0)에서 탁월한 성과를 내고 다른 메커니즘에서는 낮은 점수를 받은 채굴자가, 모든 메커니즘에서 평균적인 성과를 내는 채굴자보다 더 많은 발행량을 받을 수도 있습니다. 이는 발행 비율(emission proportion)과 상대적 성과(performance)에 따라 달라집니다.

뉴런

Bittensor 서브넷은 단순한 신경망(neural network)의 구조에서 영감을 받았으며, 각 뉴런(neuron)은 채굴자 또는 검증자 중 하나의 역할을 수행합니다. 각 뉴런은 서브넷 내에서 고유한 UID로 식별되며, 인출 및 운영을 위해 핫키-콜드키(hotkey-coldkey) 쌍과 연결되어 있습니다.

뉴런 아키텍처

서브넷 내의 뉴런은 서버-클라이언트 아키텍처(server-client architecture) 내에서 작동합니다.

- **Axon (서버):** 채굴자는 검증자로부터 데이터를 수신하고 처리하기 위해 Axon 서버를 배포합니다.
- **Dendrite (클라이언트):** 검증자는 Dendrite 클라이언트를 사용하여 채굴자에게 데이터를 전송합니다.
- **Synapse (데이터 객체):** 뉴런 간에 교환되는 데이터를 캡슐화하고 구조화합니다.

또한, **Metagraph(메타그래프)**는 서브넷 노드를 관리하기 위한 전역 디렉토리(global directory)의 역할을 하며, **Subtensor**는 뉴런을 블록체인과 연결합니다.

뉴런의 생애주기

1. **등록** → 뉴런은 PoW 또는 소각 등록(burned registration)을 통해 등록됩니다.
2. **UID 할당** → 뉴런은 서브넷 내에서 고유한 UID를 부여받습니다.
3. **면역 기간(Immunity Period)** → 뉴런은 설정된 블록 수 동안 가지치기(pruning)로부터 보호됩니다.
4. **성능 구축** → 뉴런은 순위(rank), 신뢰도(trust), 합의도(consensus), 인센티브(incentive)를 축적합니다.
5. **검증자 허가** → 스테이킹(stake) 상위 K개의 뉴런이 검증자 허가를 받습니다.
6. **가중치 설정** → 허가된 뉴런은 가중치를 설정하고 합의 과정에 참여할 수 있습니다.

7. **결속 형성** → 검증자는 성능을 기준으로 채굴자와 결속을 형성합니다.
8. **발행 분배** → 뉴런은 성능에 따라 TAO 발행량(emission)을 분배받습니다.
9. **성능 모니터링** → 뉴런의 성능은 지속적으로 평가됩니다.
10. **가지치기 위험(Pruning Risk)** → 낮은 성능의 뉴런은 새로운 등록자에 의해 교체될 위험이 있습니다.

뉴런 관리

등록 및 UID 할당

뉴런은 작업 증명(proof-of-work) 또는 소각 등록(burned registration) 방식을 통해 서브넷에 등록되며, 서브넷 내에서 고유한 UID(사용자 식별자, User ID)를 부여받습니다. 등록 과정은 추가 혹은 대체 알고리즘을 따르며, 새로운 뉴런은 서브넷을 확장하거나 기존의 저성능 뉴런을 교체하게 됩니다.

성능 지표

뉴런의 성능은 여러 지표를 통해 측정됩니다.

- **Rank(순위):** 합의(consensus) 기반 가중치 클리핑 이후의 최종 성능 점수입니다.
- **Consensus(합의도):** 절단 임계값으로 사용되는 스테이크 가중 중앙값(stake-weighted median)입니다.
- **Trust(신뢰도):** 채굴자에 대한 합의 정렬도(consensus alignment) 측정치입니다.
- **Validator Trust(검증자 신뢰도):** 검증자에 대한 합의 정렬도 측정치입니다.
- **Incentive(인센티브):** 채굴자에 대한 정규화된 보상 할당 값입니다.

검증자 허가 및 접근 제어

스테이크 가중치(stake weight)가 상위 K에 해당하는 뉴런은 검증자 허가를 부여받으며, 이를 통해 다음을 수행할 수 있습니다.

- 가중치를 설정하고 합의(consensus)에 참여합니다.
- 성능 평가를 기반으로 채굴자와 결속(bond)을 형성합니다.
- 활성 스테이크(active stake) 계산에 기여합니다.

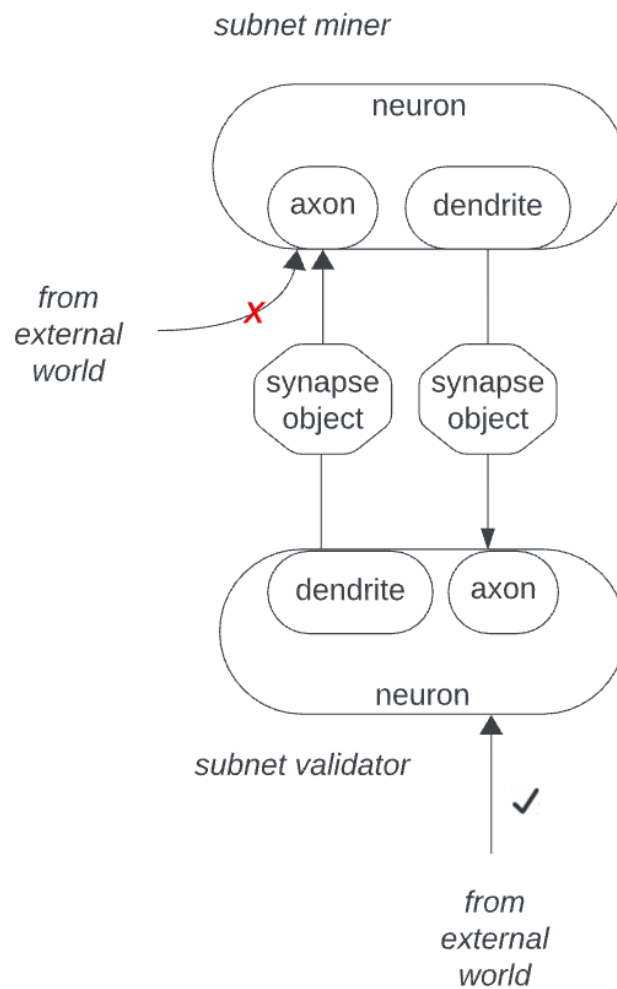
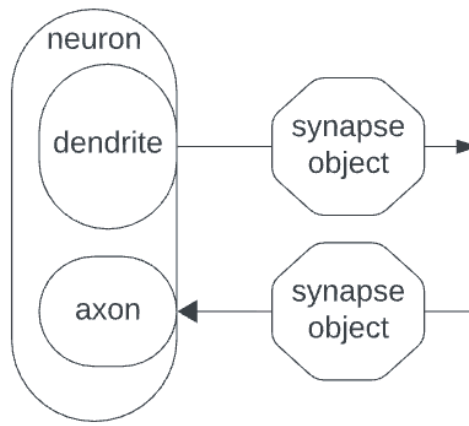
허가받은 뉴런만이 자기 자신이 아닌 타 뉴런에 대한 가중치(non-self weights)를 설정할 수 있으며, 모든 뉴런은 허가 여부와 관계없이 자기 자신에 대한 가중치(self-weight)는 설정할 수 있습니다.

뉴런 간 통신

뉴런은 다음과 같은 과정을 통해 정보를 교환합니다.

- 정보를 **Synapse(시냅스)** 객체에 캡슐화합니다.
- 서버(**Axon**) 및 클라이언트(**Dendrite**) 네트워크 요소를 인스턴스화하고, 이 서버-클라이언트(Axon-Dendrite) 프로토콜을 사용하여 Synapse 객체를 교환합니다.

아래 다이어그램을 참조하십시오.



Axon (서버)

Bittensor API의 axon 모듈은 FastAPI 라이브러리를 사용하여 API 서버를 생성하고 실행합니다. 예를 들어, 서브넷 채굴자가 다음과 같이 호출할 때:

```
axon = bt.axon(wallet=self.wallet, config=self.config)
```

axon이라는 이름의 API 서버가 서브넷 채굴자 노드 상에 생성됩니다. 이 API 서버는 서브넷 검증자로부터 들어오는 Synapse 객체를 수신하며, 즉 axon은 서브넷 채굴자를 대신하여 서비스를 시작하게 됩니다. 마찬가지로, 서브넷 채굴자 코드에서는 서브넷 검증자로부터 들어오는 Synapse 객체를 수신하기 위해 axon API를 사용하여 API 서버를 생성해야 합니다.

Dendrite (클라이언트)

Axon은 **서버(server)** 인스턴스입니다. 따라서, 서브넷 검증자는 서브넷 채굴자에 존재하는 Axon으로 정보를 전송하기 위해 자체적으로 **dendrite** 클라이언트를 인스턴스화해야 합니다. 예를 들어, 서브넷 검증자가 다음 코드 조각을 실행할 때:

```
responses: List[bt.Synapse] = await self.dendrite(
    axons=axons,
    synapse=synapse,
    timeout=timeout,
)
```

서브넷 검증자는 다음을 수행합니다.

- 자체적으로 dendrite 클라이언트를 인스턴스화합니다.
- synapse 객체를 서브넷 채굴자에 연결된 여러 axons에게 전송합니다.
- timeout이 만료될 때까지 응답을 대기합니다.

Synapse (데이터 객체)

Synapse(시냅스)는 데이터 객체(data object)입니다. 서브넷 검증자(subnet validator)와 서브넷 채굴자(subnet miner)는 정보를 교환하기 위한 주요 수단으로 Synapse 데이터 객체를 사용합니다. Synapse 클래스는 Pydantic 데이터 검증 라이브러리의 BaseModel 클래스를 상속받습니다.

예를 들어, [텍스트 프롬프트 서브넷\(Text Prompting Subnet\)](#)에서는 서브넷 검증자가 PromptingSynapse라는 Synapse 객체를 생성하며, 이 객체는 roles, messages, 그리고 completion의 세 가지 필드를 포함합니다. roles와 messages 필드는 이 Prompting 데이터 객체의 초기화 시 서브넷 검증자에 의해 설정되며, 이후에는 변경할 수 없습니다. 세 번째 필드인 completion은 변경 가능(mutable)합니다. 서브넷 채굴자가 서브넷 검증자로부터 이 Prompting 객체를 수신하면, 채굴자는 completion 필드를 업데이트합니다. 그 후, 서브넷 검증자는 이 업데이트된 completion 필드를 읽습니다.

메타그래프 (The Metagraph)

Metagraph(메타그래프)는 서브넷의 현재 상태에 대한 종합적인 정보를 포함하는 데이터 구조(data structure)입니다.

서브넷의 메타그래프를 조사하면, 해당 서브넷 내 모든 노드(뉴런)에 대한 상세 정보를 확인할 수 있습니다. 서브넷 검증자는 자신이 속한 서브넷의 모든 채굴자를 파악하기 위해 먼저 해당 서브넷의 메타그래프와 동기화(sync)해야 합니다. 메타그래프는 Bittensor API를 사용하여 프로그래밍 방식으로 검사할 수 있습니다.

발행

발행은 비트텐서의 경제적 중심으로, 새롭게 생성된 TAO와 서브넷 고유의 알파 토큰(alpha token)을 네트워크에 기여하는 참여자에게 지속적으로 분배하는 과정입니다. 이 참여자에는 [채굴\(mining\)](#), [검증\(validation\)](#), [스테이킹\(staking\)](#), [서브넷 생성\(subnet-creation\)](#) 등이 포함됩니다.

2단계 발행 구조 이해

비트텐서의 발행 시스템은 공정하고 시장 기반의 분배를 보장하기 위해 서로 연동되는 두 단계로 구성되어 있습니다.

1단계: 서브넷으로의 투입

매 블록마다 새로운 유동성(liquidity)이 서브넷 풀(subnet pool)로 투입됩니다. 시장 성과가 높은 알파 토큰을 보유한 서브넷일수록 더 많은 TAO를 유입받게 되며, 이는 혁신을 위한 경쟁적 시장 구조를 형성합니다.

2단계: 참여자에 의한 분배

매 템포(tempo) (360 블록, 약 72분)마다 누적된 보상이 Yuma Consensus를 통해 참여자에게 분배됩니다. 이 합의 메커니즘은 참여자의 성과를 평가하고, 그에 따라 보상의 비율을 결정합니다.

이 2단계 구조는 안정성(stability)과 반응성(responsiveness)을 동시에 확보합니다. 보상은 서서히 축적되지만, 실제 분배는 명확한 가치 창출(performance)에 기반해 이루어집니다.

투입 (Injection)

발행의 첫 번째 단계는 서브넷 풀(subnet pool)로의 유동성 투입(liquidity injection)입니다. 유동성은 각 서브넷의 토큰 가격을 다른 서브넷 토큰들과 비교한 비율에 따라 서브넷별로 투입됩니다. 이는 가장 높은 가치를 지닌 서브넷에서의 개발을 장려하도록 설계되었습니다.

각 블록마다 다음과 같은 과정이 수행됩니다:

- TAO가 서브넷의 **TAO 준비금(TAO reserve)**으로 투입됩니다.
- 알파(Alpha)가 서브넷의 **알파 준비금(alpha reserve)**으로 투입됩니다.
- 알파가 **알파 유통 잔액(alpha outstanding)**으로 할당되어, 참여자들이 분배(extract)할 수 있도록 준비됩니다.

TAO 준비금 투입

서브넷의 TAO 준비금 투입량은 해당 서브넷의 알파 토큰 가격을 비트텐서 내 모든 서브넷의 알파 토큰 가격 합으로 나눈 비율에 따라 계산됩니다.

알파 준비금 투입

알파 역시 토큰의 가격 비율에 따라 투입되며, 이로써 서브넷의 유동성 풀(liquidity pool) 성장이 알파 토큰의 가격에 영향을 미치지 않도록 보장합니다.

알파 유통 잔액 투입

각 블록마다 일부 유동성이 참여자(검증자, 채굴자, 스테이커, 서브넷 생성자)에게 분배될 수 있도록 별도로 설정됩니다. 이때 블록당 분배되는 양은 해당 서브넷 i 의 알파 발행률(alpha emission rate) $\Delta\alpha_i$ 와 동일합니다.

분배 (Extraction)

각 템포(tempo) (360 블록)가 종료될 때, 해당 템포 동안 누적된 알파(Alpha)는 다음 비율에 따라 네트워크 참여자에게 분배(extraction)됩니다:

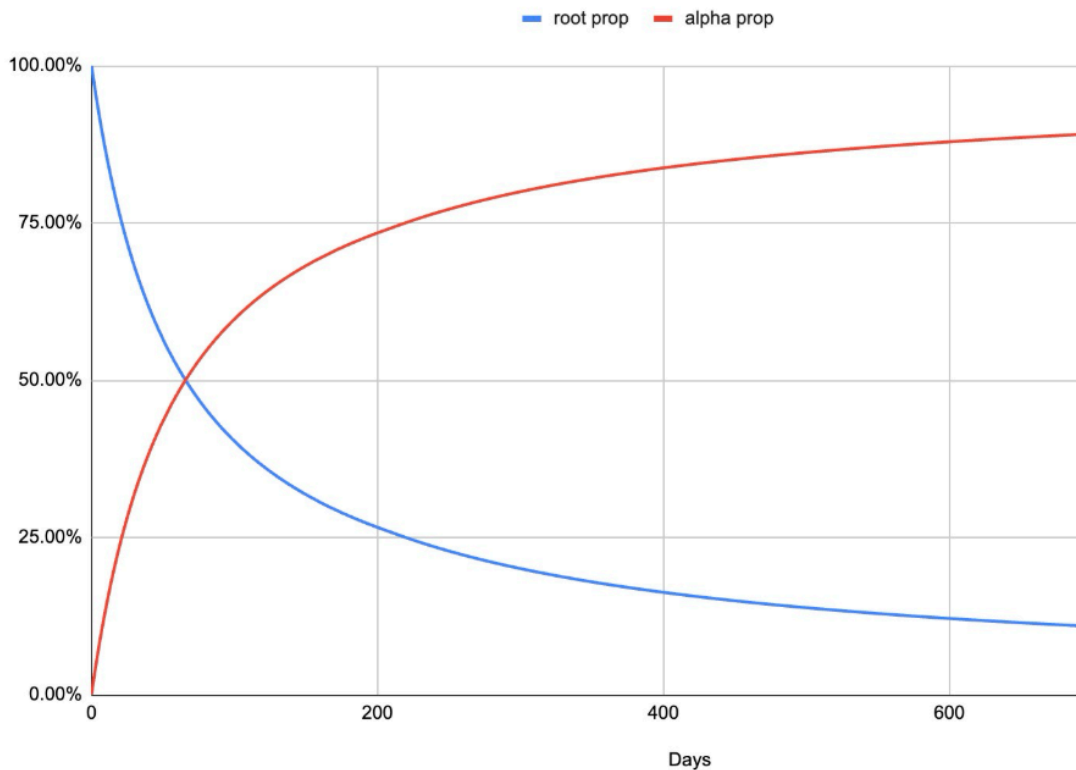
1. **18%** — 서브넷 소유자(subnet owner)에게 지급
2. **41%** — 채굴자(miners)에게 지급
 - 개별 채굴자에 대한 분배 비율은 [Yuma Consensus: Miner Emissions](#)에 의해 결정됩니다.
3. **41%** — 검증자(validators)와 그들의 스테이커(stakers)에게 지급
 - i. 검증자 및 채굴자에 대한 배분은 [Yuma Consensus: Validator Emissions](#)에 의해 결정됩니다.
 - ii. 검증자는 해당 배분 중 자신의 몫을 추출합니다.
 - iii. 스테이커에게는 검증자의 토큰 보유 비율에 따라 TAO와 알파가 발행됩니다.
TAO 발행은 서브넷의 유동성 풀을 통해 알파 발행량의 일부를 TAO로 교환(swap)함으로써 이루어집니다.

비트텐서 토큰 이코노미 진화에 대한 사항

Dynamic TAO가 시작될 때, 유통 중인 알파(alpha)는 존재하지 않으므로 검증자의 스테이크 가중치는 전적으로 그들의 TAO 스테이크 비율에 의해 결정됩니다.

그러나 매 블록마다 TAO보다 훨씬 더 많은 양의 알파가 유통에 방출됩니다. 그 결과, 시간이 지남에 따라 전체 유통량에서 TAO 대비 알파의 비중이 점차 증가하게 되며, 특정 서브넷(subnet)에서 검증자의 상대적 가중치는 Subnet Zero에서의 TAO 스테이크 비율에 대한 알파 스테이크 비율(alpha stake share)에 더욱 의존하게 됩니다.

네트워크 내에서 알파가 스테이크 권한의 과반수를 더 빠르게 차지하도록 하기 위해, 검증자의 스테이크 가중치에 대한 TAO 스테이크의 기여도는 TAO weight라고 불리는 글로벌 파라미터에 의해 감소됩니다. 현재 이 값은 약 18%로 설정될 예정이며, 약 100일 내에 TAO와 전체 알파 간의 가중치 균형을 달성하기 위한 것입니다.



비트텐서에서의 지수 이동 평균(EMA)

지수 이동 평균(EMA)은 시간에 따라 변화하는 값(예: 토큰 가격)을 추적하기 위한 수학적 기법입니다. EMA는 오래된 데이터 포인트의 가중치를 지수적으로 감소시키는 가중 이동 평균으로, 최근에 값이 *가장 많이 머물렀던* 위치를 반영하는 신호를 추출하여 빠르고 대부분 무작위적인 변동의 지속적인 노이즈를 안정화하거나 “평활화(smoothing)”합니다.

비트텐서는 발행(emission) 과정 중 두 가지 핵심 동적 값(dynamical values)을 평활화하기 위해 EMA를 사용합니다.

- 각 서브넛으로의 발행량은 각 템포(tempo)마다 서브넛 가격의 EMA-평활화 표현에 의해 결정됩니다. 이는 가격 변동성이나 의도적인 조작으로부터 발행 메커니즘을 보호합니다.
- 각 서브넛의 참여자에게 발행되는 보상은 검증자-채굴자 간의 순간적인 결속 강도(instantaneous bond strength)의 EMA에 의해 결정됩니다. 이는 즉각적인 합의가 아닌 최종 합의(eventual consensus)를 기준으로 혁신에 대한 공정한 보상을 보장하는 데 중요한 역할을 합니다.

수학적 정의

주어진 시점에서 변화하는 값의 EMA는 현재 값과 직전 시점의 EMA의 가중 평균으로 결정됩니다. 이때 사용되는 파라미터 계수를 *평활화 계수* (smoothing factor)라고 하며, α 로 표시됩니다.

$$EMA^{(t)} = \alpha \times current + (1 - \alpha) \times EMA^{(t-1)}$$

알파(α) 파라미터는 EMA가 변화에 얼마나 빠르게 반응하는지를 제어합니다.

- 작은 α (예: 0.01):** 반응 속도가 매우 느리고 안정성이 높으며, 큰 변화가 반영되기까지 많은 기간이 필요합니다.
- 큰 α (예: 0.5):** 반응 속도가 빠르지만 안정성이 낮으며, 새로운 정보를 빠르게 반영합니다.
- $\alpha = 1$:** 평활화가 전혀 없는 즉각적인 반응을 의미합니다.

참고:

여기서 사용되는 알파(α) 파라미터는 서브넛별 통화를 나타내는 “alpha” 용어와는 구별되며, 서로 관련이 없습니다.

서브넛 가격 발행 평활화

EMA 평활화를 이용한 이 메커니즘은 발행량이 가격 변동에 매우 느리게 반응하도록 하여, 네트워크의 경제 모델이 가격 조작에 영향을 받지 않도록 보호합니다.

작동 방식: 가격 EMA는 새로운 서브넛이 성숙한 서브넛보다 더 느리게 가격이 적응하도록 하기 위해, 동적 알파 계산(dynamic alpha calculation)을 사용합니다.

$$\alpha = \frac{basealpha \times blockssincestart}{blockssincestart + halvingblocks}$$

검증자-마이너 결속 평활화

이 평활화 함수는 검증자와 채굴자 간의 관계가 점진적으로 발전하도록 하여, 갑작스러운 조작을 방지하면서 유망한 마이너를 일찍 발견한 검증자에게 보상이 주어지도록 합니다.

기본 결속 EMA (Liquid Alpha 비활성화 상태)

기본 모드: 모든 검증자-마이너 쌍에 대해 단일 α 사용.

- 기본값:** $\alpha \approx 0.1$ (10%)
- 반응 시간:** 주요 변화에 대해 약 7-22 블록 (약 1-4분)
- 공식:**
시점 t 에서 검증자 i 의 마이너 j 에 대한 결합(BondEMA)의 EMA는 순간 결합 값(instantaneous bond)과 이전 시점의 BondEMA의 α -가중 평균으로 계산됩니다.

$$BondEMA^{(t)} * ij = \alpha \times InstantBond * ij + (1 - \alpha) \times BondEMA_{ij}^{(t-1)}$$

고급 결속 EMA (Liquid Alpha 활성화 상태)

합의 기반 모드: 합의 정렬(consensus alignment)에 따라 검증자-채굴자 쌍별로 동적으로 변화하는 α 를 사용합니다.

- α 범위:** α_{low} 와 α_{high} 사이에서 동적으로 변화 (기본값: 0.7~0.9)
- 시그모이드 기울:** α_{low} 에서 α_{high} 로의 전환 속도를 제어 (기본값: 1000)
- 개별 알파:** 각 검증자-마이너 쌍은 고유한 α 값을 가집니다.
- 반응 시간:** 합의 정렬 정도에 따라 1~13 블록 (약 12초~2.6분)

Yuma 컨센서스

소개

Yuma Consensus (YC)는 비트덴서 내에서 Subtensor 상에서 온체인으로 실행되는 핵심 알고리즘적 프로세스입니다. 그 역할은 검증자들이 채굴자의 성능을 평가하여 제출한 순위를 기반으로 검증자와 채굴자 각각의 보상(emission)을 계산하는 것입니다.

각 서브넷의 검증자들은 자신이 평가한 각 채굴자의 작업 가치를 순위화한 가중치(weight) 벡터를 주기적으로 제출합니다. YC 알고리즘은 이 순위 행렬을 두 개의 보상 벡터(emissions vectors)로 변환합니다 — 하나는 채굴자용, 다른 하나는 검증자용입니다. 이 보상 벡터들은 참여자의 성과에 따라 보상을 분배합니다.

이 알고리즘은 신뢰도가 높은 검증자의 입력에 더 큰 가중치를 부여하도록 설계되어, 신뢰성이 낮은 검증 신호를 무시합니다. 신뢰할 수 없는 가중치 설정을 배제함으로써, YC는 검증자들이 다른 검증자들의 평가가 최종적으로 수렴할 것을 예측하는 일관되고, 신속하며, 정직한 평가를 제공하도록 노력하게끔 유도합니다. YC는 또한 채굴자들이 검증자 커뮤니티 전체로부터 가능한 가장 높은 종합 평가를 받기 위해 노력하도록 장려합니다.

클리핑(Clipping)

클리핑은 부정확한 채굴자 평가를 처벌하도록 설계되었으며, 특히 특정 채굴자를 유리하게 만들기 위해 합의를 조작하는 담합(collusion) 형태의 패턴에 대해 이를 방지합니다.

이를 위해, 가장 신뢰받는 검증자(스테이크로 측정됨)의 판단이 기준으로 사용됩니다. 이 기준을 초과하는 가중치는 클리핑 되어, 해당 채굴자와 검증자 모두 그 부분에 대한 보상을 받지 못하게 됩니다.

이러한 클리핑 메커니즘은 검증자들이 채굴자를 과도하게 또는 담합적으로 고평가하는 잘못된 평가로부터 시스템을 보호합니다.

채굴자 j 와 서브넷의 검증자 집합 $\mathbb{V}$ 에 대한 벤치마크 $\overline{W}_j$ 를 계산하기 위해, 모든 검증자 $i \in \mathbb{V}$ 의 가중치 W_{ij} 를 수집하고, 검증자의 스테이크 S_i 에 따라 이를 정렬한 다음, 전체 스테이크 중 적어도 κ 비율이 지지하는 최대 가중치 수준 w 를 찾습니다.

$$\overline{W}_j = \arg \max_w \left(\sum_{i \in \mathbb{V}} S_i \cdot W_{ij} \geq w \geq \kappa \right)$$

검증자의 원래 가중치 W_{ij} 가 $\overline{W}_j$ 를 초과할 경우, 그 값은 다음과 같이 클리핑(clipping) 됩니다.

$$\overline{W}_{ij} = \min(W_{ij}, \overline{W}_j)$$

채굴자 보상

채굴자 보상은 집계된 순위(aggregate ranking)에 기반합니다. 이는 검증자들의 순위를 그들의 스테이크에 따라 가중합한 결과로, 여기서 $\overline{W}_{ij}$ 는 클리핑 이후의 가중치입니다.

$$R_j = \sum_{i \in \mathbb{V}} S_i \cdot \overline{W}_{ij}$$

각 채굴자 j 의 서브넷 내 채굴자 보상 비율 M_j 는 서브넷 전체 채굴자 보상(각 서브넷 전체 보상의 41%) 중 자신이 차지하는 총합 순위 비율에 해당합니다. 여기서 $\mathbb{M}$ 은 서브넷의 채굴자 집합입니다.

$$M_j = \frac{R_j}{\sum_{k \in \mathbb{M}} R_k}$$

합의에서 벗어난 결속에 대한 페널티

만약 검증자의 채굴자 평가가 과도하게 높을 경우, 해당 검증자는 페널티를 받습니다. 즉, 검증자 i 가 채굴자 j 에 대해 제출한 가중치 W_{ij} 가 채굴자의 합의 평가값 $\overline{W}_j$ 를 초과할 경우, 해당 검증자의 본드(bond) 가치는 계수 β 만큼 감점됩니다.

결속 가중치 $\widehat{W}_{ij}$ 는 다음과 같이 정의됩니다.

$$\widehat{W}_{ij} = (1 - \beta)W_{ij} + \beta\overline{W}_{ij}$$

결속 메커니즘

검증자 i 가 채굴자 j 에게 가지는 **즉시 결속(instant bond)** ΔB_{ij} 는 검증자 i 의 스테이크 S_i 에 i 가 채굴자 j 에게 설정한 결속 가중치를 곱하고, j 가 속한 서브�트의 모든 검증자 집합 $\mathbb{V}$ 에 대한 총 결속 가중치로 정규화한 값과 같습니다.

$$\Delta B_{ij} = \frac{S_i \cdot \widehat{W}_{ij}}{\sum_{k \in \mathbb{V}} S_k \cdot \widehat{W}_{kj}}$$

이후 이 값은 **지수이동평균(Exponential Moving Average, EMA)** 형태의 본드로 갱신됩니다.

$$B_{ij}^{(t)} = \alpha, \Delta B_{ij} + (1 - \alpha), B_{ij}^{(t-1)}$$

합의 근처에 머무르는 검증자일수록 더 강한 EMA 본드를 형성하며, 그 결과 더 많은 보상을 얻습니다. 반면, 채굴자의 성능을 과대평가하려는 모든 시도는 페널티를 받습니다. EMA는 검증자 행동의 급격한 변화를 완화하고, 합의와의 일관된 정렬을 유지하도록 유도합니다.

검증자 보상

각 검증자 i 의 보상 지분 V_i 는 검증자 보상 총량(각 서브넷 전체 보상의 41%) 중 자신이 보유한 몫으로, 이는 해당 검증자가 채굴자들에게 형성한 모든 결속(bond)의 합에 채굴자의 총 보상(emissions)을 가중하여 계산됩니다.

$$V_i = \sum_{j \in \mathbb{M}} (B_{ij} \times M_j)$$

YC 3가 비트텐서를 더욱 공정하게 만드는 방식

YC3는 비트텐서의 합의 메커니즘의 차세대 진화 버전입니다. 이 버전은 혁신적인 채굴자를 신속히 인식한 검증자들에게 보상이 돌아가도록 보상(emission) 최적화를 수행합니다.

Yuma Consensus 이해하기

모든 비트텐서 서브�트의 중심에는 근본적인 과제가 존재합니다. 즉, 다양한 디지털 자원이나 서비스를 포함할 수 있는 작업의 가치를 어떻게 공정하게 분배할 것인가 하는 문제입니다. 비트텐서는 이를 분산된 판단 문제로 접근합니다. 검증자는 전체 커뮤니티의 심판 역할을 하며, 이들의 신뢰도는 위임받은 총 스테이크(stake) 양으로 측정됩니다. 이들이 각 서브넛에서 자산과 서비스를 생산하는 채굴자의 성과를 평가한 결과가 해당 채굴자에게 분배되는 보상(emissions)을 결정합니다.

Yuma Consensus는 Bittensor가 제시한 합의 솔루션입니다. 검증자들은 자신이 속한 서브넛에서 채굴자들이 수행한 작업의 품질을 지속적으로 평가하며, 각 검증자의 평가 신뢰도는 커뮤니티로부터 받은 스테이크 양에 비례하여 결정됩니다. 게으르거나 부정직한 검증자는 부정확한 평가를 제출함으로써 보상(emission)을 잃게 되며, 이로 인해 커뮤니티는 스테이크를 더 신뢰할 수 있는 검증자에게 이전하게 됩니다. 따라서 특정 검증자에 대한 커뮤니티의 신뢰(스테이크로 표현됨)는 그 검증자가 평가한 채굴자들이 시간이 지나면서 얼마나 많은 보상을 받는가와 직접적으로 연결됩니다. 이러한 구조를 통해 검증자들은 정직하고 성실하게 행동할 유인을 가지게 되며, 채굴자들은 가능한 한 최고의 디지털 자원(commodities)을 생산하기 위해 지속적으로 노력하게 됩니다.

각 검증자는 자신이 평가한 채굴자들의 순위를 제출합니다. 이후 알고리즘은 이 순위 데이터를 분석하여 어떤 검증자들이 가장 신뢰할 수 있는 평가를 제공하는지를 식별합니다. 다른 검증자들이 결국 최고의 채굴자로 인정하게 될 채굴자들을 일관되게 잘 예측한 검증자는 더 높은 영향력과 보상을 받습니다. 반면, 평가가 부정확하거나 시의적절하지 못한 검증자는 손해를 보게 됩니다.

이 시스템은 시간에 따라 검증자와 채굴자 사이에 “결속(bond)”을 형성합니다. 검증자가 채굴자의 뛰어난 작업을 꾸준히 인식하면, 그 결속은 강화되어 양측 모두에게 더 나은 보상으로 이어집니다. 이로써 검증자들이 세심하고 정직하며, 선제적으로 판단하도록 유도하는 긍정적 피드백 루프가 만들어 집니다.

그러나 이전 버전의 Yuma Consensus에는 몇 가지 한계가 있었습니다. 특히 잠재력 있는 신생 채굴자를 조기에 인식하는 문제와 소규모 스테이크 검증자에게 공정성을 보장하는 부분에서 개선이 필요했습니다. Yuma Consensus 3는 이러한 문제들을 해결합니다.

YC3란

YC3는 기존 Yuma Consensus 메커니즘을 대체하는 버전으로, 여러 가지 핵심적인 문제를 해결하면서도 이전 버전과의 하위 호환성을 유지합니다. 대부분의 서브넷 운영자들은 별도의 변경을 가할 필요가 없으며, 이 업그레이드는 원활하게 적용되도록 설계되었습니다.

YC3의 가장 중요한 혁신은 결속별 EMA 스케일링(per-bond EMA scaling)입니다. 이제 각 검증자-채굴자 결속 쌍마다 고유한 조정률(α 값)을 가지며, 더 이상 모든 본드에 동일한 비율을 적용하지 않습니다. 이를 통해 개별 관계가 성능 및 합의 편차에 따라 서로 다른 속도로 발전할 수 있습니다.

타임라인

- 현재: YC3는 이미 활성화되어 있으며 여러 서브넷에서 도입 중입니다.
- 강제적인 마이그레이션 기한은 없습니다 — 각 서브넷은 준비가 되었을 때 YC3를 채택할 수 있습니다.

주요 개선 사항

1. 공정한 검증자 대우

- **해결된 사항:** 소규모 검증자들이 단순히 규모가 작다는 이유로 반올림 오류로 인해 불이익을 받았습니다.
- **YC3의 해결책:** 본드 값이 이제 고정소수점 연산(fixed-point arithmetic)으로 계산되고, 저장 효율성을 위해 u16 형식으로 변환됩니다. 이를 통해 0-65535 범위 내에서 저장 효율성을 유지하면서도 정밀한 소수 단위 관계를 표현할 수 있습니다.

2. 결속 정밀도 문제

- **해결된 사항:** 16비트 정수 정밀도가 결속 누적에 충분하지 않아, 일부 검증자가 채굴자에게 가중치를 주었음에도 결속을 받지 못하는 경우가 있었습니다.
- **YC3의 해결책:** 고정소수점 연산을 활용한 정밀도 향상으로 모든 검증자가 적절한 결속 할당을 받도록 보장합니다.

3. 결속 업스케일링 및 감쇠

- **해결된 사항:** 합의값이 0일 때 결속 업스케일링이 불공정한 분배를 유발했습니다.
- **YC3의 해결책:** 향상된 EMA 스무딩(EMA smoothing)이 적용된 결속 메커니즘을 통해 더 예측 가능하고 공정한 보상 분배를 보장합니다.

4. 조기 인식 보상

- **새로운 기능:** 유망한 채굴자를 다른 이들보다 먼저 인식한 검증자는 이제 초기에 결속을 누적하기 시작할 수 있습니다.
- **기술적 구현:** 알파 시그모이드(alpha sigmoid) 함수가 검증자의 현재 가중치와 네트워크 합의 간의 차이를 기반으로 조정률을 계산합니다.

5. 향상된 조정 가능성(Enhanced Tunability)

- **새로운 기능:** 추가적인 파라미터를 통해 서브넷 운영자가 자신의 합의 메커니즘을 보다 정교하게 조정할 수 있습니다.
- **하위 호환성 유지:** 기존 서브넷은 기본 설정으로 계속 정상적으로 작동합니다.

토큰 브릿징

- TAO를 Substrate 스타일의 지갑(SS58)과 비트텐서 EVM의 Ethereum 유형 지갑 간에 이동하는 방법
- 비트텐서 EVM과 다른 EVM 체인 간에 vTAO를 토큰 브리지로 사용하는 방법

비트텐서 EVM의 스마트컨트랙트는 비트텐서 블록체인 상에서만 실행되며, Ethereum 블록체인에서는 실행되지 않습니다.

비트텐서 체인에서 Substrate 및 EVM 지갑 간 유동성 전송

TAO는 비트텐서 네트워크의 네이티브 토큰입니다. 이는 Subtensor(비트텐서의 블록체인) 상에 존재하며, Substrate 프레임워크 위에 구축되어 있습니다. 따라서 TAO는 일반적으로 Substrate 형식의 ss58 지갑에 보관되며, 비트텐서 Python SDK 또는 BTCLI를 통해 Subtensor 블록체인의 extrinsic을 실행할 수 있습니다.

TAO를 Ethereum 유형의 h160 지갑으로 전송하면, 동일한 토큰이지만 다른 계정 형식으로 표현되어 비트텐서의 EVM 레이어에서 사용할 수 있습니다.

TAO는 Substrate 지갑과 EVM 지갑 간에 여러 방법으로 상호 이동할 수 있습니다.

- [H160 → SS58으로 TAO 전송](#)
- [SS58 → H160으로 TAO 전송](#)
- [tao.app/bridge](#) 사용
- OTF(OpenTensor Foundation)의 EVM 브리지 사용: bridge.bittensor.com

vTAO를 이용한 다른 EVM 체인 브릿지

vTAO는 Subtensor EVM에서 사용할 수 있는 리퀴드 스테이킹(Liquid-Staked) TAO 토큰으로, [tao.app/bridge](#)를 통해 발행됩니다.

- vTAO는 스테이킹 컨트랙트에 TAO를 예치함으로써 발행되며, 이후 환율에 따라 TAO로 다시 상환(redeem)할 수 있습니다.
- vTAO 잔액은 동일하게 유지되지만, 스테이킹 보상에 따라 컨트랙트에 잠긴 실제 TAO는 점차 증가합니다.
- vTAO는 지원되는 EVM 체인 간에 브리지(bridge)될 수 있습니다.

비트텐서의 거래 수수료

블록체인의 상태를 변경하는 대부분의 외부 트랜잭션(extrinsic transactions)은 소량의 가중치 기반 수수료(weight-based fee)가 부과됩니다. 스테이킹(staking) 및 언스테이킹(unstaking) 작업에는 가중치 기반 수수료뿐만 아니라 거래 규모의 0.05% 금액 기반 수수료(amount-based fee)도 발생합니다. 블록체인의 상태를 조회하는 것은 항상 무료입니다.

가중치 기반 거래 수수료

비트코인의 많은 외부 트랜잭션은 가중치 기반 수수료가 부과됩니다. Polkadot 기반 체인(Subtensor, 즉 비트코인의 레이어 1 블록체인)에서 *가중치(weight)*는 연산 시간(compute time)의 척도입니다.

수수료 세부사항

- **지불 원천:** 기본적으로 송신자의 TAO 가용 잔고(free balance)에서 차감됩니다. 특정 extrinsic의 경우 TAO가 부족하면, 체인은 대신 Alpha로 수수료를 부과합니다.
- **단위:** 기본적으로 TAO입니다. 수수료가 Alpha로 지불될 경우, TAO 수수료 금액은 현재 Alpha 가격을 기준으로 Alpha로 변환됩니다 (슬리피지 없음).
- **유동성 영향:** 수수료는 *재활용(recycled)*되어 TotalIssuance에서 차감됩니다.

Alpha 대체 결제(fallback)

Alpha를 TAO로 스왑하여 수수료를 부과하는 extrinsic의 경우, 송신자의 TAO 잔액이 가중치 기반 거래 수수료를 감당할 수 없으면 체인은 자동으로 Alpha로 수수료를 부과하는 대체 결제 메커니즘(fallback mechanism)을 사용합니다. TAO와 Alpha 잔액이 모두 예상 수수료를 충당하기에 부족한 경우, 거래는 검증에 실패하며 메모에 포함되지 않습니다. 수수료가 Alpha로 지불될 때, TAO 수수료는 현재 Alpha 가격을 기준으로 슬리피지 없이 Alpha로 변환됩니다.

스테이킹 및 언스테이킹 스왑 수수료

앞서 설명한 가중치 기반 수수료 외에도, 스테이킹(staking) 및 언스테이킹(unstaking) 작업에는 거래된 유동성 양의 일정 비율에 따른 수수료가 부과됩니다. 스테이킹을 서브넷 간에 이동(transfer, swap 또는 move)을 통해 할 때 0.05%의 수수료가 적용됩니다. 단, 동일한 서브넷 내에서 이동하는 경우에는 추가 수수료가 부과되지 않으며, 가중치 기반 수수료만 적용됩니다.

수수료 세부사항

- **비율:** 0.05%
- **스테이킹 시:** 스테이킹 금액에서 TAO로 수수료 지불
- **언스테이킹 시:** 언스테이킹 금액에서 Alpha로 수수료 지불

지갑, 콜드키(coldkey), 핫키(hotkey)

비트코인에서 지갑(wallet)은 사용자의 신원을 증명하고, 거래를 서명하며, TAO에 접근하고, 서브넷에서의 스테이크를 관리하기 위한 도구입니다.

지갑과 키

다양한 지갑 애플리케이션(wallet applications)이 존재하지만, 지갑의 핵심은 콜드키(coldkey)와 핫키(hotkey)로 불리는 하나 이상의 암호화 키 쌍입니다. 각각은 실제로 암호화 키 쌍(key-pair)으로 구성되어 있으며, 비공개 키(private key)와 공개 키(public key)로 이루어집니다. 공개 키는 수학적으로 비공개 키로부터 파생되며, 비공개 키는 거래 서명 및 비밀 복호화를 가능하게 하는 핵심 정보로, 암호학적 신원 증명 수단 역할을 합니다. 이것은 분산 원장(distributed ledger)이나 블록체인과 같은 탈중앙화된 신뢰 없는 시스템의 일반적인 특징으로, 개인 키의 도난이나 손실은 곧 접근 권한의 완전한 상실(복구 불가)을 의미합니다.

비트코인에서 콜드키와 핫키는 서로 다른 작업에 사용됩니다. 요약하자면, 핫키는 마이닝(mining) 및 검증(validation)을 위해 사용되며, 콜드키는 그 외의 모든 작업에 사용됩니다. 만약 사용자가 채굴자도 검증자도 아니라면 핫키가 필요하지 않지만, 검증자 및 채굴자는 각자의 핫키 공개 키로 식별됩니다.

콜드키의 비공개 키(coldkey private key)는 TAO 잔액 전송, 스테이크 관리, 서브넷 관리 및 거버넌스 관련 작업, 그리고 핫키 관리 등 고보안 작업을 승인할 때 필요합니다. 반면, 핫키의 비공개 키(hotkey private key)는 서브넷 내에서 채굴자가 요청을 처리하도록 허가하거나, 검증자가 채굴자에게 요청을 전송하고 가중치를 블록체인에 제출할 때 사용됩니다.

콜드키 공개 키(coldkey public key)는 네트워크 상에서 지갑을 고유하게 식별하며, 거래 주소로 사용됩니다. TAO 또는 Alpha 스테이크의 소유권을 다른 지갑으로 이전할 때, 송신자는 수신자의 공개 키와 자신의 비공개 키만 있으면 됩니다.

지갑과 지갑 애플리케이션

*지갑(wallet)*이라는 용어는 두 가지 의미로 혼용될 수 있으므로 주의가 필요합니다.

- **암호화 지갑(cryptographic wallet):** 하나 이상의 암호화 키 쌍으로 구성되어 신원을 증명하고, 거래 서명 및 상대방의 거래에서 참조될 수 있도록 하는 구조입니다. 이 의미에서 지갑은 자산 접근을 통제하고 공개 신원 역할을 하는 **콜드키**와 거의 동일한 개념입니다.

- **지갑 애플리케이션(wallet application):** 사용자의 장치에서 실행되어 키를 입력하고 블록체인과 상호작용할 수 있도록 하는 소프트웨어입니다. 비트텐서에서 공식적으로 지원하는 지갑 애플리케이션은 다음과 같습니다.
 - 모바일용 비트텐서 지갑 앱: bittensor.com/wallet
 - Crucible 지갑 — 서버넷 간 동적 TAO 스테이킹을 위한 자동 할당기(auto-allocator)를 갖춘 TAO 지갑으로, Ledger와 완전히 통합되어 있습니다.
 - [Polkadot 브라우저 확장 프로그램](#) — Polkadot Vault와 함께 사용할 수 있습니다.
 - 비트텐서 브라우저 확장 지갑 — Ledger 하드웨어 지갑과 호환됩니다. 자세한 내용은 [Ledger 하드웨어 지갑 사용법](#)을 참조하십시오.
 - 비트텐서 지갑 브라우저 확장 프로그램을 사용하면 [TAO.app](#)을 통해 스테이크를 관리할 수 있습니다.
 - 비트텐서 Python SDK — 보안 비트텐서 Wallet 모듈을 포함합니다.
 - 비트텐서 CLI(btcli) — 내부적으로 비트텐서 Wallet 모듈을 사용합니다.

일부 사용자는 하나 이상의 암호화 지갑을 가질 수도 있습니다. 모든 경우에 하나 이상의 암호화 지갑을 여러 지갑 애플리케이션에 로드할 수 있습니다. 모든 애플리케이션에서 로그아웃하거나 장치를 폐기하더라도, 블록체인 상의 시드 문구(seed phrase)를 통해 암호화 지갑은 복구할 수 있습니다.

지갑 애플리케이션마다 기능 수준이 다릅니다.

- 모바일 앱과 Chrome 확장 프로그램은 TAO 잔액의 스테이킹 및 전송이 가능하지만, 핫키 관리나 고급 기능은 포함되지 않습니다.
 - Chrome 확장 프로그램은 하드웨어 지갑과 호환되어 보안성이 높습니다. 이는 노트북을 콜드키 워크스테이션(coldkey workstation)으로 사용하는 것을 의미합니다.
 - 모바일 앱은 안전한 스마트폰을 콜드키 워크스테이션으로 사용하는 것을 전제로 합니다.
- btcli와 SDK는 핫키 관리 및 기타 고급 기능을 제공합니다. 이들은 노트북을 콜드키 워크스테이션으로 사용하는 것이 필요합니다.

시드 문구(Seed Phrase, Mnemonic)

시드 문구(Seed Phrase) 또는 *니모닉(Mnemonic)*, *복구 문구(Recovery Phrase)*는 지갑의 암호화 키 쌍이 생성될 때 함께 만들어지는 일련의 단어(최소 12개 이상)로 구성됩니다. 이 시드 문구는 콜드키의 비공개 키(coldkey private key)를 복구할 수 있으며, 따라서 암호화 지갑에 대한 접근 권한을 오프라인으로 보관하거나 지갑 애플리케이션에 암호화 지갑을 가져올 수 있는 사람 친화적인 방식입니다. 비트텐서 지갑을 다룰 때 가장 중요한 운영상의 목표 중 하나는 시드 문구를 분실하거나 유출하지 않는 것입니다.

지갑 애플리케이션

공개 키(public key) 및/또는 비공개 키(private key)와 다양한 방식으로 상호작용할 수 있는 애플리케이션이 존재합니다.

무허가성 지갑 애플리케이션

bittensor.com/scan을 방문하여 콜드키의 공개 키를 입력하면, 해당 지갑의 공개 정보를 확인할 수 있습니다. 이 경우 브라우저는 일종의 비허가형(permissionless) 지갑 애플리케이션으로 작동하여, 지갑의 공개 데이터를 표시할 수 있습니다.

스테이킹 애플리케이션

콜드키를 안전하게 다룰 수 있는 여러 애플리케이션이 존재합니다. 이는 기본적으로 콜드키를 디스크에 암호화되지 않은 상태로 저장하지 않고도 애플리케이션에 로드할 수 있음을 의미합니다. 이론적으로, 암호화 비밀번호가 없는 장치 접근이 이루어지더라도 키를 탈취하는 것은 불가능합니다. 그러나 주의할 점은, 장치에서 콜드키를 사용하는 것 자체가 공격자에게 다른 형태의 공격 벡터를 제공할 수 있다는 것입니다.

보안성이 검증된 지갑 애플리케이션에는 다음이 포함됩니다.

- **비트텐서 모바일 지갑 앱** — Opentensor Foundation 제공: bittensor.com/wallet
- **Chrome 확장 프로그램** — Opentensor Foundation 제공, Ledger 하드웨어 지갑과 호환됩니다. 자세한 내용은 [Ledger 하드웨어 지갑 사용법](#)을 참조하십시오.
- **Crucible Labs의 Crucible 지갑** — 서버넷 간 동적 TAO 스테이킹을 위한 자동 할당기(auto-allocator)를 갖춘 TAO 지갑으로, Ledger와 완전 통합되어 있습니다.

콜드키(Coldkey)

btcli에서 콜드키(coldkey)는 지갑 이름(wallet name)과 동일한 개념입니다. 예를 들어, btcli 명령어의 --wallet.name 옵션은 항상 <coldkey> 값만을 허용하며, --wallet.hotkey 옵션은 <hotkey> 값만을 허용합니다. 이는 콜드키가 온체인에서 여러 개의 핫키(hotkey)에 대한 권한(permission)과 소유권(ownership)을 보유하기 때문입니다. 따라서 지갑 이름은 콜드키에 할당됩니다.

핫키와의 관계성

콜드키는 핫키 없이도 존재할 수 있으며, 여러 개의 핫키를 가질 수도 있습니다. 예를 들어, 서버넷을 생성하거나, 스테이크를 위임(delegate)하거나, 단순히 잔액을 보유하는 등의 행위를 할 때는 콜드키만으로 충분합니다. 그러나 서버넷에서 검증을 하거나 채굴을 하려면, 이 콜드키와 페어링된 핫키(hotkey)가 반드시 필요합니다.

목적

콜드키는 TAO 전송(transfer), 스테이킹(staking), 언스테이킹(unstaking) 등 잔액(balance)에 영향을 미치는 모든 작업에 필요합니다. 또한 콜드키는 새로운 핫키를 생성하고 등록하는 작업, 서버넷 관리, 거버넌스 관련 기능에서도 반드시 사용됩니다.

암호화

콜드키는 항상 디스크에 암호화된(encrypted) 형태로 저장되며, 접근을 위해서는 암호화 비밀번호(encryption password)가 필요합니다.

존재적 예치금 (Existential Deposit)

존재적 예치금(Existential Deposit)은 지갑(즉, 콜드키)에 존재해야 하는 최소 TAO 금액을 의미합니다. 만약 지갑의 잔액이 이 최소 금액 아래로 떨어지면, 해당 지갑 계정은 비활성화(deactivated)되며 남아 있는 TAO는 소멸됩니다. 비트텐서 네트워크의 모든 지갑에 대해 이 값은 500 RAO로 설정되어 있습니다.

핫키(Hotkey)

핫키는 서브넷(subnet)에서 채굴자(miner) 또는 검증자(validator)로 등록하는 데 사용됩니다.

콜드키와의 관계성

하나의 콜드키에 여러 개의 핫키를 생성하여 연결할 수 있습니다. 그러나 사용자가 서브넷 내에서 검증 또는 채굴을 수행할 때는 해당 서브넷에서 핫키를 통해 식별되므로 콜드키는 노출되지 않습니다. 같은 서브넷 내에서는 하나의 핫키를 여러 UID에 사용할 수 없습니다 — 즉, 각 UID는 고유한 핫키를 필요로 합니다. 다만, 다른 서브넷 간에는 동일한 핫키를 재사용할 수 있습니다.

목적

핫키는 비트텐서 네트워크에서 다음과 같은 일반적인 운영 작업에 사용됩니다.

- 트랜잭션 서명
- 서브넷 채굴자 및 검증자 등록 및 실행
- 검증자인 경우, TAO 보유자들이 TAO를 전송할 수 있도록 자신의 핫키를 지정
- TAO 보유자인 경우, 보유한 TAO를 검증자 대리인(validator-delegate)의 핫키에 위임

기 관련 FAQ

하나의 콜드키에 여러 핫키를 연결할 수 있나요?

네, 가능합니다. 채굴자나 검증자는 하나의 콜드키를 사용해 여러 서브넷에서의 마이닝 및 검증을 위한 다수의 핫키를 관리할 수 있습니다.

동일한 서브넷 내의 여러 UID에 같은 핫키를 사용할 수 있나요?

아니요. 하나의 서브넷 내에서는 핫키 하나당 하나의 UID만 사용할 수 있습니다. 하지만 서로 다른 서브넷의 UID에는 같은 핫키를 재사용할 수 있습니다.

TAO를 핫키로 전송할 수 있나요?

기술적으로는 가능하지만 절대 권장되지 않습니다. 이는 설계 의도에 반하는 행위이며, 시도할 경우 자금을 영구적으로 잃을 위험이 있습니다. TAO 전송은 항상 콜드키로만 이루어져야 하며, 콜드키의 공개 키가 지갑의 거래 주소 역할을 합니다.

콜드키는 서브넷과 연관되어 있나요?

네, 그렇습니다. 서브넷이 생성될 때 netuid는 해당 서브넷 등록 수수료를 지불한 콜드키에 연결됩니다. 이 콜드키는 서브넷의 구성 설정(하이퍼파라미터)을 변경하는 데 필요하며, 서브넷 발행량의 18%가 이 콜드키로 배분됩니다.

서브넷 검증자와 채굴자의 발행량은 어디에 입금되나요?

검증자와 채굴자는 자신이 활동하는 서브넷의 알파(α) 토큰으로 발행 보상을 받습니다. 이 보상은 해당 검증자 또는 채굴자의 핫키에 스테이크(stake) 형태로 쌓이며, 이후 언스테이킹(unstake)을 통해 해당 콜드키의 TAO 잔액으로 전환할 수 있습니다.

TAO를 검증자의 핫키에 위임하려면 어떻게 하나요?

TAO 보유자는 자신이 가진 TAO를 해당 서브넷의 알파(α) 토큰으로 교환(stake)하여, 특정 서브넷의 검증자 핫키에 스테이킹 또는 위임할 수 있습니다. 즉, 검증자의 핫키가 항상 스테이킹의 수신자 역할을 합니다.

스테이킹/위임

TAO 보유자는 자신이 보유한 유동성의 일부 또는 전부를 검증자에게 스테이킹(stake)할 수 있습니다. 이 과정은 위임(delegation)이라고도 불리며, 검증자를 지원하는 역할을 합니다. 이는 검증자의 서브넷 내 총 스테이크(자신의 스테이크 + 위임받은 스테이크)가 검증자의 합의 권한(consensus power)과 발행 보상(emission share)을 결정하기 때문입니다. 검증자(또는 대리인, delegate)가 자신의 테이크(take)를 차감한 뒤 남은 보상은, 검증자에게 스테이킹한 비율에 따라 스테이커(또는 위임자)에게 분배됩니다.

검증자/대리인은 자신의 테이크(take) 비율을 설정할 수 있습니다. 기본값은 18% 입니다.
명령어 예시: `btcli sudo set-take`. 위임자(nominator)의 최소 스테이킹 금액은 0.1 TAO입니다.

스테이킹은 항상 서브넷 단위로(local to a subnet) 이루어집니다. 각 서브넷은 자체적인 자동화된 시장조성자(AMM, Automated Market Maker)를 운영하며, 이는 두 토큰의 준비금(reserve)을 유지하여 유동성을 원하는 양만큼 교환할 수 있도록 합니다. 가격은 자동으로 계산되며, 시장 조성자는 서브넷 내 거래의 유동성을 보장합니다.

각 서브넷은 TAO 준비금과 해당 서브넷의 통화인 알파(α) 준비금을 보유하고 있습니다. 스테이크는 항상 α 토큰 단위로 보관됩니다. TAO 보유자는 특정 서브넷의 검증자 핫키에 스테이킹하게 되며, 동일 검증자의 다른 서브넷 핫키에 스테이킹하는 것은 서로 독립적입니다.

스테이킹 시

1. 먼저, 사용자의 TAO 스테이크가 서브넷의 AMM 풀 내 **TAO 준비금**으로 예치됩니다.
2. 이후 AMM 알고리즘은 최신 환율을 이용해 추가된 TAO에 상응하는 **α 단위의 스테이크 양**을 계산합니다.
계산된 α 는 풀의 알파 준비금에서 차감되어 검증자의 핫키로 전송됩니다.
3. 검증자의 핫키는 이 α 를 보유하게 되며, 해당 서브넷의 모든 핫키 스테이크의 총합은 $\alpha_{outstanding}$ 으로 불립니다.

언스테이킹 시

1. 사용자가 `btcli stake remove` 명령을 실행하고 언스테이킹할 α 토큰 양을 지정하면, 해당 α 는 검증자의 핫키에서 차감되어 서브넷 풀의 α 준비금으로 되돌아갑니다.
2. AMM 알고리즘은 최신 환율을 이용해 추가된 α 에 상응하는 **TAO 단위**를 계산합니다.
3. 계산된 TAO는 서브넷 풀의 TAO 준비금에서 차감되어, 해당 TAO 보유자의 콜드키로 전송됩니다.

스테이크는 항상 알파 단위로 표현됩니다.

Dynamic TAO 환경에서, 루트 서브넷(Root Subnet)에 보관된 스테이크를 제외한 모든 스테이크는 서브넷 고유의 α 단위로 표현됩니다. 루트 서브넷의 스테이크는 τ 단위로 표시됩니다.

예시

한 검증자가 자신의 **800 TAO**를 보유하고 있다고 가정합니다.

세 명의 위임자가 다음과 같이 스테이킹합니다.

- 위임자 1: 100 TAO
- 위임자 2: 70 TAO
- 위임자 3: 30 TAO

검증자의 유효 스테이크는 자신의 스테이크와 위임받은 스테이크의 합으로 계산됩니다.

$$\text{delegated stake} = 100\tau + 70\tau + 30\tau = 200\tau$$

$$\text{total stake} = \text{self-stake} + \text{delegated stake} = 800\tau + 200\tau = 1000\tau$$

스테이커에 대한 발행 보상은 위임된 스테이크 비율에 비례합니다.

$$\text{emission for staker } x \text{ from validator } V = \frac{\text{stake} * x}{\sum *i \in V'stakers(\text{stake}_i)}$$

- **위임자 1:** 전체 위임 TAO의 50%

$$\frac{100\tau}{100\tau + 70\tau + 30\tau} = 0.5$$

- **위임자 2:** 전체 위임 TAO의 35%

$$\frac{70\tau}{100\tau + 70\tau + 30\tau} = 0.35$$

- **위임자 3:** 전체 위임 TAO의 15%

$$\frac{30\tau}{100\tau + 70\tau + 30\tau} = 0.15$$

검증자는 전체 스테이크의 80%를 보유하므로, 총 발행량의 80%를 유지하며, 또한 위임된 스테이크에서 발생한 발행 보상의 18%를 추가로 가져갑니다.

결과적으로:

- **검증자가 받는 총 발행량:**

$$0.8 + 0.2 \times 0.18 = 0.836 = 83.6$$

• 각 위임자의 수익:

- 위임자 1: $(1 - 0.8) \times (1 - 0.18) \times 0.5 = 0.082 = 8.2$
- 위임자 2: $(1 - 0.8) \times (1 - 0.18) \times 0.35 = 0.0574 = 5.74$
- 위임자 3: $(1 - 0.8) \times (1 - 0.18) \times 0.15 = 0.0246 = 2.46$

위임자는 곧 위임 권한자입니다 (A Nominator Is a Delegating Authority)

위임자는 TAO 자금을 보유하고 있지만 검증 작업을 수행하지 않고, 대신 비트텐서 네트워크에 투자하는 개인 또는 기관을 의미합니다.

비트텐서에서의 채굴

서브넷 선택

비트텐서의 모든 채굴은 서브넷 내에서 이루어집니다. 각 서브넷은 고유한 목적에 따라 디지털 자원(digital commodities)을 생산하며, 각 서브넷 생성자는 검증자가 채굴자의 작업을 평가하는 기준으로 사용할 인센티브 메커니즘을 정의합니다. 검증자는 서브넷의 인센티브 메커니즘에 따라 채굴자의 성과를 평가하며, 이 점수가 해당 서브넷의 발행량(emission) 중 채굴자에게 배정되는 비율을 결정합니다.

비트텐서에서의 채굴은 비트코인(Bitcoin)이나 기타 블록체인의 채굴과는 상당히 다릅니다. 이는 능동적(active)이며, 창의적(creative)이고, 경쟁적(competitive)입니다. 서브넷 채굴자가 되기 위해서는 자신의 전문 분야와 하드웨어 접근성을 고려하여, 자신에게 가장 적합한 서브넷을 조사하는 것이 필수적입니다.

채굴자 등록

채굴자로 참여하려면, 먼저 서브넷에 핫키(hotkey)를 등록하여 해당 서브넷에서 UID를 발급받아야 합니다.

등록에는 TAO 비용이 발생하며, 이는 마지막 등록 이후 경과된 시간에 따라 동적으로 변동됩니다. 메인 체인의 서브넷에서 UID 슬롯을 확보하면, 이 TAO는 매몰 비용(sunk cost)으로 간주되며 반환되지 않습니다.

하나의 서브넷(Subnet 1 제외)은 최대 64개의 검증자 UID와 192개의 채굴자 UID(총 256개)를 가질 수 있습니다.

등록이 완료되면, 지갑의 일부인 핫키(hotkey)가 UID 슬롯의 소유자가 됩니다.

채굴자 등록 해제

채굴자가 검증자로부터 낮은 가중치(weight, 평점)를 받아 낮은 발행량(emission)을 기록할 경우, 등록이 해제될 수 있습니다. 일반적인 서브넷은 서브넷당 256개의 UID 슬롯을 가지며, 이 중 최대 64개는 검증자가 사용할 수 있습니다. 각 템포(tempo)가 끝날 때, 가장 낮은 순위의 슬롯이 해당 핫키에서 등록 해제(deregistered)되어 새 등록자에게 할당됩니다.

일반적으로 서브넷은 256개의 UID 슬롯을 가지며, 그 중 기본적으로 64개의 슬롯이 검증자에게 할당될 수 있습니다.

따라서 192개의 UID가 채굴자에게 남게 됩니다. 단, 서브넷에 64명 미만의 검증자가 존재할 경우, 채굴자가 그 빈 슬롯을 차지할 수 있습니다.

각 템포마다, 발행량(emission)에 기반한 가지치기 점수(pruning score)가 가장 낮고 immunity_period가 종료된 뉴런(채굴자 또는 검증자)은 새로 등록된 뉴런에게 UID를 넘겨줄 위험이 있습니다.

등록 해제는 발행량(emission)에 의해 결정됩니다.

서브넷은 등록 해제 시 채굴자와 검증자를 구분하지 않습니다.

체인은 오직 발행량(가지치기 점수)만을 기준으로 판단합니다.

서브넷에 새로운 등록이 발생할 때마다, 발행량이 가장 낮은 뉴런이 등록 해제됩니다.

채굴자를 위한 자동 스테이킹

자동 스테이킹(auto staking)은 채굴자가 선택한 검증자에게 채굴 수익을 자동으로 스테이킹하여,복리 스테이킹(compound staking) 과정을 수동 개입 없이 간소화할 수 있도록 하는 기능입니다.

자동 스테이킹 기능은 채굴자가 자신의 채굴 발행량(emission)을 자동으로 스테이킹할 대상 검증자(destination validator)를 설정할 수 있게 합니다. 이를 통해 수동으로 스테이킹 작업을 수행할 필요 없이, 채굴 보상이 지속적으로 네트워크에 재투자되도록 합니다.

자동 스테이킹이 설정되면, 채굴자가 서브넷 참여를 통해 발행량을 획득할 때마다 그 보상이 지정된 검증자에게 자동으로 스테이킹됩니다. 이 기능은 채굴자가 수익을 얻는 동시에 지분을 자연스럽게 늘릴 수 있도록 하며, 반복적인 수동 스테이킹 작업을 수행할 필요가 없게 합니다.

비트텐서에서의 검증

비트텐서의 모든 검증 작업은 서브넷 내에서 수행됩니다. 각 서브넷은 고유한 목적에 따라 디지털 자원(digital commodities)을 생산하며, 서브넷 생성자는 검증자가 채굴자의 작업을 평가하는 데 사용할 인센티브 메커니즘을 정의합니다. 검증자의 역할은 이 인센티브 메커니즘을 채굴자에게 적용하여 그들의 성과를 평가하고, 이 평가 결과(가중치)를 비트텐서 블록체인에 제출하는 것입니다. 검증자가 평가한 채굴자들의 성과 점수는 Yuma Consensus 알고리즘에 따라 서브넷의 발행량 중 각 채굴자에게 배분되는 비율을 결정합니다.

작동 원리

비트텐서 블록체인의 각 서브넷은 최대 256개의 활성 노드(active node)를 지원하며, 각 노드는 고유한 UID 슬롯을 할당받습니다. 이 중 발행량(emission) 기준 상위 64개 노드만이 기본적으로 검증자로 참여할 수 있습니다. 즉, 서브넷에 64명의 검증자가 존재한다는 것은 상위 64개의 노드가 필수 조건을 충족하고 검증자로 참여하기로 선택했다는 의미입니다.

검증자로 자격을 얻으려면 검증자 허가증(validator permit)이 필요합니다. 이 허가증은 상위 64개 노드에만 부여되며, 검증자는 이를 통해 `btcli weights commit` 명령어나 SDK의 `set_weights` 함수를 사용하여 채굴자 평가를 제출할 수 있습니다.

자격 요건

특정 서브넷에서 검증자 허가증(validator permit)을 받기 위해서는 다음 조건을 충족해야 합니다.

- 해당 서브넷에 UID를 발급받은 핫키(hotkey)가 등록되어 있어야 합니다.
- 서브넷 내에서 최소 1,000의 스테이크 가중치(stake-weight)를 보유해야 하며, 여기에는 다른 지갑의 콜드키(coldkey)로부터 위임된 스테이크도 포함됩니다. 검증자의 서브넷 내 스테이크 가중치는 해당 검증자의 알파 스테이크(α)와 TAO 스테이크(τ)의 조합으로 계산됩니다. 단, TAO 스테이크에는 `tao_weight` 파라미터(현재 값: 0.18)가 곱해집니다.

$$\text{Validator stake weight} = \alpha + 0.18 \times \tau$$

- 또한, 해당 서브넷 내에서 발행량 기준 상위 64개 노드 중 하나여야 합니다.

검증자 등록

검증자로 참여하려면, 먼저 서브넷에 핫키(hotkey)를 등록하여 해당 서브넷에서 UID를 발급받아야 합니다.

기본적으로 하나의 서브넷은 최대 64개의 활성 검증자 UID를 가질 수 있습니다. 등록이 완료되면, 지갑의 일부인 핫키가 해당 UID 슬롯의 소유자가 됩니다.

등록 명령어는 다음과 같습니다:

```
btcli subnet register --netuid <desired netuid> --wallet.name <wallet name> --wallet.hotkey <validator hotkey>
```

검증자 등록 해제

검증자는 채굴자와 마찬가지로, 발행량(emission)이 낮을 경우 등록이 해제될 수 있습니다. 그러나 검증자 등록 해제는 채굴자 해제보다 더 복잡한 절차를 거칩니다. 이는 활성 검증자가 반드시 서브넷 내 상위 64개 노드에 포함되어야 하며, 새로 등록된 노드가 즉시 “가지치기(pruning)”를 수행할 수 없기 때문입니다.

검증자가 발행량 기준 상위 64개 노드 아래로 떨어지거나, 필수 스테이크 가중치(총 1000 stake weight) 미만으로 내려가면 검증자 허가증(validator permit)을 잃게 됩니다. 그러나 이는 즉시 등록 해제를 의미하지는 않습니다. 검증자가 허가증을 잃고 발행량을 회복할 수 없는 상태가 지속되면, 결국 해당 검증자는 서브넷 내에서 가장 낮은 발행량을 가진 노드가 되어 등록 해제 대상이 됩니다.

등록 해제는 **모든 256개의 UID 슬롯이 꽉 찬 서브넷에서만 발생합니다.**

UID 슬롯이 남아 있는 서브넷에 새로운 등록이 이루어질 경우, 새로 등록된 뉴런(neuron)은 비어 있는 슬롯을 차지하게 됩니다.

각 템포(tempo)마다, 발행량(emission)에 기반한 가지치기 점수(pruning score)가 가장 낮고 immunity_period가 만료된 뉴런(채굴자 또는 검증자)은 새로 등록된 뉴런에게 UID를 넘겨줄 위험이 있습니다.

등록 해제는 **발행량(emission)에 기반합니다.**

서브넷은 등록 해제 시 채굴자와 검증자를 구분하지 않습니다. 체인은 오직 발행량(가지치기 점수)만을 기준으로 판단합니다. 새로운 등록이 발생할 때마다, 발행량이 가장 낮은 뉴런이 등록 해제됩니다.

스테이크 확보

검증자의 합의 가중치(consensus weight)와 발행량(emission)은 해당 검증자의 핫키 스테이크 가중치(stake weight)에 따라 결정됩니다. 검증자는 자신의 TAO를 검증자 핫키에 직접 스테이킹할 수 있으며, 또는 핫키를 공개하여 다른 사용자로부터 스테이킹을 받을 수도 있습니다. 모든 지갑의 콜드키(coldkey)는 임의의 핫키에 스테이킹할 수 있으며, 그에 따른 발행량 보상을 받을 수 있습니다.

검증자 허가증

검증자 허가증은 어떤 뉴런이 서브넷 내 검증 활동에 참여할 수 있는지를 제어합니다. 이 시스템은 스테이크 기반(stake-weighted)으로 작동하며, 높은 스테이크를 가진 뉴런만이 합의에 영향을 미칠 수 있도록 보장합니다.

허가증 계산 알고리즘

검증자 허가증은 매 에포크(epoch)마다 다음 절차를 통해 계산됩니다.

- 스테이크 필터링 (Stake Filtering):** 최소 1000의 스테이크 가중치를 가진 뉴런만 고려됩니다.
- 상위 K 선택 (Top-K Selection):** 스테이크 가중치 기준 상위 K개의 뉴런이 검증자 허가증을 부여받습니다. (일반적으로 상위 64개)
- 동적 갱신 (Dynamic Updates):** 현재 스테이크 분포에 따라 매 에포크마다 허가증이 재계산됩니다.

접근 제어 및 보안

검증자 허가증은 다음과 같은 핵심 네트워크 기능을 제어합니다.

- 가중치 설정 (Weight Setting):** 허가증이 있는 뉴런만이 자기 자신이 아닌 다른 노드의 가중치를 설정할 수 있습니다.
- 합의 참여 (Consensus Participation):** 허가증이 있는 뉴런만이 Yuma Consensus에 기여할 수 있습니다.
- 결속 관리 (Bond Management):** 뉴런은 검증자 허가증을 유지하는 동안만 결속 관계를 유지할 수 있습니다.
- 활성 스테이크 (Active Stake):** 허가증이 있는 뉴런만이 활성 스테이크 계산에 포함됩니다.

허가증 요건

검증자 허가증을 얻기 위해 뉴런은 다음 조건을 충족해야 합니다.

- 최소 스테이크:** 최소 1000의 스테이크 가중치가 필요합니다. ($\alpha + 0.18 \times \tau$)
- 상위 K 순위 (Top-K Ranking):** 스테이크 가중치 기준 상위 K개의 뉴런 중 하나여야 합니다.
- 활성 상태:** 서브넷 내에서 활성 상태를 유지해야 합니다.

허가증 수명주기 및 결속 관리

검증자 허가증을 잃으면, 해당 허가증과 연결된 결속(bond)이 삭제됩니다. 이러한 구조는 현재 자격 요건을 충족하는 검증자만이 합의에 영향을 미칠 수 있도록 보장합니다.

사용자 유동성 포지션(Uniswap)

개요

유동성 포지션(Liquidity Position, LP) 기능은 사용자가 특정 서브넷의 α 토큰에 대해 지정된 가격 범위 내에서 거래 유동성을 제공할 수 있도록 합니다. 이 시스템은 Uniswap V3의 집중형 유동성(concentrated liquidity) 모델을 기반으로 하며, 제공자는 거래 활동으로부터 수수료를 획득할 수 있습니다.

모든 TAO 보유자는 유동성 포지션(LP)을 생성하여 서브넷의 스테이킹/언스테이킹 유동성을 제공하고 서브넷 토큰의 가격 안정성에 기여할 수 있습니다. 사용자가 정의된 가격 범위 내에서 스테이킹 및 언스테이킹을 수행할 때, 유동성 포지션은 수수료를 누적하며, LP 생성자는 이후 이 수수료를 자신의 지갑으로 인출할 수 있습니다.

서브넷 생성자는 자신의 서브넷에서 유동성 포지션 기능을 활성화하거나 비활성화할 수 있습니다.

유동성 포지션 vs. 스테이킹

TAO를 검증자에게 스테이킹할 때, 이는 해당 검증자가 서브넷의 합의 메커니즘(consensus mechanism)에 참여하도록 “투표”하는 것과 같습니다. 검증자의 총 스테이크(자신의 스테이크 + 위임받은 스테이크)는 네트워크 내에서 그들의 발행량(emission) 비율과 영향력을 결정합니다.

스테이커(Staker)는 자신이 보유한 스테이크에 대해 발행 보상(emission)을 각 템포(tempo)마다 분배받습니다.

반면, 유동성 포지션(LP)은 다른 사용자가 해당 포지션의 정의된 가격 범위 내에서 스테이킹 또는 언스테이킹을 수행할 때 거래 수수료(fees)를 얻습니다.

즉, 서브넷의 거래 풀에 유동성을 제공함으로써 사용자는 TAO와 해당 서브넷의 Alpha 토큰 간 거래를 가능하게 하고, 이를 통해 서브넷 내 유동성을 높이며 서브넷 토큰의 시장 가격을 안정화하는 역할을 수행합니다.

동적 토큰 구성

유동성 포지션(LP)은 TAO, 알파(alpha), 또는 두 토큰 모두를 보유할 수 있습니다. 이는 LP 생성 시 지정된 가격 범위와 서브넷의 현재 토큰 가격 간의 상대적 위치에 따라 결정됩니다.

이 구성은 LP를 생성할 때 필요한 토큰 비율뿐만 아니라, 유동성을 제거할 때 얻는 토큰의 종류와 양을 의미합니다. 이는 거래가 실행되는 블록 시점에서 토큰 가격이 LP의 가격 범위 내 어디에 위치하느냐에 따라 달라집니다.

유동성 포지션의 수명 주기

포지션 생성

LP를 생성하기 위해 사용자는 유동성(liquidity) 파라미터를 지정합니다. 이 값은 TAO와 알파 토큰의 일정 조합으로 변환됩니다. TAO는 사용자의 콜드키에서 차감되며, 알파 토큰은 LP가 생성된 핫키에서 차감되어 LP 안에 잠금(lock) 상태로 보관됩니다.

포지션 수정

LP 생성자는 기존 포지션에 유동성을 추가하거나 제거할 수 있습니다. 유동성을 추가할 때 필요한 토큰의 양, 또는 유동성을 제거할 때 반환되는 토큰의 양은 LP 생성 시와 동일한 수식에 의해 계산됩니다.

수수료 누적

사용자가 포지션의 가격 범위 내에서 TAO와 Alpha 간의 스왑을 수행할 때마다 수수료가 발생합니다.

블록체인은 각 포지션의 수수료를 다음 요소에 따라 계산합니다:

- 스테이킹/언스테이킹된 TAO 및 Alpha의 양
- 거래가 포함된 가격 범위를 가진 다른 LP들 대비 해당 포지션의 상대적 유동성

수수료 분배

수수료는 발행 보상(emission)처럼 각 템포(tempo)마다 자동으로 분배되지 않습니다. 대신, 수수료는 사용자가 직접 유동성을 인출할 때만 지갑으로 분배됩니다.

- **포지션을 수정할 때(유동성 추가 또는 제거 시):**
누적된 모든 수수료가 자동으로 수집되어 지갑으로 전송됩니다.
- **포지션을 완전히 제거할 때:**
누적된 모든 수수료가 포지션 내 토큰과 함께 수집됩니다.

즉, 사용자는 수익으로 발생한 수수료를 수령하기 위해 직접 포지션을 관리해야 하며, 포지션 수정 또는 제거 작업을 수행하기 전까지 수수료는 해당 포지션 내에 잠긴 상태로 유지됩니다.

포지션 제거

포지션이 제거(destroyed/removed)될 때, 포지션 내의 유동성은 현재 서버넷 가격과 포지션이 정의된 가격 범위의 관계에 따라 토큰으로 환산됩니다. 변환이 완료되면 포지션은 시스템에서 삭제됩니다.

Liquidity 파라미터

liquidity 파라미터는 유동성 공급자(LP)를 정의하는 값이지만, TAO 토큰이나 Alpha 토큰(또는 두 토큰의 합계)을 의미하는 것은 아닙니다. 이는 Uniswap V3의 집중 유동성(concentrated liquidity) 모델에서 비롯된 수학적 스케일링 인자(scaling factor)로, LP를 생성할 때 핫키(Alpha)와 콜드키(TAO)에서 차감되는 토큰 수량을 계산하는 데 사용됩니다. 실제로 잡기는 TAO와 Alpha의 양은 아래 의사 코드(pseudocode)로 표현된 `to_token_amounts()` 함수에 의해 계산됩니다.

거버넌스 개요

비트텐서의 거버넌스 프로토콜은 네트워크의 관리 권한을 초기 단계의 재단 중심 구조에서 점차 커뮤니티 중심의 자율적 운영 체계로 이전시키는 역할을 합니다.

이 탈중앙화된 관리 체계로의 전환의 첫 단계는 양원제 입법 구조(bicameral legislature)의 도입입니다. 이 단계에서 3인 위원회(Triumvirate)가 제안을 작성하고, 상원(Senate)이 이를 승인하는 구조로 운영됩니다.

3인위원회 구성원은 Opentensor Foundation의 직원으로 구성되며, 상원은 상위 K명의 대표 검증자(delegate hotkeys)로 구성됩니다.

제안

제안은 다른 extrinsic들을 캡슐화한 형태로 존재하며, 다음의 두 조건이 모두 충족될 때만 실행됩니다.

1. 상원(Senate)으로부터 (50% + 1) 이상의 승인을 획득해야 합니다.
2. 3인 위원회의 구성원 중 한 명이 해당 제안을 종결(close)해야 합니다.

위 조건은 제안이 실행되기 위해 상원이 반드시 과반수 합의(majority consensus)에 도달해야 함을 보장합니다.

보안

거버넌스 프로토콜이 도입되기 이전에는, 네트워크 내의 모든 관리 작업(예: 하이퍼파라미터 변경, 새로운 서브넷 생성, 체인 업그레이드 등)이 **sudo**로 알려진 단일 특권 키(privileged key)를 통해서만 수행될 수 있었습니다. 만약 sudo의 비공개 키(private key)가 탈취된다면, 악의적인 공격자는 네트워크를 장악하여 모든 특권 extrinsic을 실행할 수 있는 위험이 존재했습니다.

하지만 거버넌스 프로토콜 하에서는, 공격자가 네트워크를 조작하려면 3인 위원회 구성원 1명 이상과 타협하고 동시에 상원의 과반수 의석을 통제해야 합니다. 따라서 단일 키 탈취에 의한 위험이 구조적으로 완화되었습니다.

예시

다음 상황을 가정해 봅시다.

- 3인 위원회에는 세 개의 의석이 있으며, 구성원은 Alice, Bob, Charlie이다.
- 상원(Senate)에는 선출된 세 명의 구성원 Dave, Eve, Ferdie가 참여하고 있다.

3인 위원회

Bob은 새로운 서브넷(subnet)에 대한 새로운 컨셉을 가지고 있으며, 이를 Bittensor 네트워크에 배포하고자 한다. Bob은 다음과 같은 콜데이터를 포함한 제안(proposal)을 작성한다.

```
SubtensorModule.SudoAddNetwork(netuid, tempo, modality)
```

그리고 이 트랜잭션을 네트워크에 전송하여 제안을 브로드캐스트한다.

상원

- Dave, Eve, Ferdie는 모두 지명된(delegate) 핫키(hotkey)를 보유하고 있으며, 각각 네트워크 전체 스테이크의 2% 이상을 개별적으로 통제한다.
- 이들은 `btcli`를 사용하여 제안 내용과 승인 시 실행될 호출 데이터(calldata)를 확인할 수 있다.
- Dave와 Ferdie는 새로운 서브넷에 찬성하여 제안을 승인(approve)했다.
- 반면, Eve는 이 개념에 동의하지 않아 제안을 반대(disapprove)했다.

상원(Senate)은 언제나 최대 열두 명의 구성원을 가질 수 있지만, 실제로 열두 자리가 모두 채워져 있을 필요는 없다. 현재 상원 규모가 세 명일 경우, 승인 기준(approval threshold)은 두 명의 승인이다. Dave와 Ferdie가 모두 찬성했으므로, 이제 3인 위원회의 구성원 중 한 명이 이 제안을 실행할 수 있다.

종결

Alice는 상원이 제안을 통과시킨 것을 확인한 뒤, `close extrinsic`을 실행하여 제안 내의 콜데이터를 실제로 실행한다.

이제 Bittensor에는 새로운 서브넷이 생성되었으며, Alice, Bob, 또는 Charlie는 이 서브넷에서 하이퍼파라미터를 변경하거나, 등록 허용 여부를 조정하거나, 기존에 sudo 개인 키로만 수행할 수 있던 설정들을 직접 제안(propose)할 수 있게 된다.

상원

상원(Senate)은 제안에 참여하고, 네트워크 총 스테이크의 상당 부분을 통제하는 대표자(delegate)들의 집합입니다.

네트워크의 모든 구성원은 자신이 위임(stake delegation)한 상원 구성원을 통해 대표됩니다. 즉, TAO 보유자는 자신이 선택한 대표(delegate)의 이해관계를 기반으로 네트워크 의사결정에 간접적으로 참여할 수 있습니다. 이 구조는 네트워크의 모든 참여자가 자신의 의견을 반영할 수 있도록 합니다.

참여 요건

상원(Senate)에 참여하기 위해서는 콜드키(coldkey)가 다음 조건을 모두 충족해야 합니다.

- 하나 이상의 서브넷(subnetwork)에 핫키-콜드키 쌍으로 등록되어 있어야 합니다.
- 해당 핫키의 스테이크 값이 네트워크 총 스테이크의 2% 이상이어야 합니다. (자체 스테이킹 또는 위임 포함)
- 상원 참여 의사를 명시적으로 선택해야 합니다.

위의 요건을 모두 충족한 콜드키-핫키 쌍은 3인 위원회가 생성한 모든 제안에 대해 투표할 수 있습니다.

만약 상원 12석이 모두 채워진 상태에서 새로운 대표가 참여를 희망한다면, 그 대표는 네트워크 내에서 더 많은 스테이크를 보유하고 있을 경우 가장 낮은 스테이크 보유자를 대체하게 됩니다.

제안 조회

현재 상원에서 검토 중인 제안은 누구나 열람할 수 있습니다. 이는 별도의 권한이 필요 없는 공개 요청(unpermissioned request)입니다.

투표

상원의 구성원(Senators)은 다음 명령어를 사용하여 투표할 수 있습니다. 먼저 `btcli sudo proposals` 명령어를 통해 제안의 해시(hash)를 확인해야 합니다.

이후 제안 해시를 입력하면, 승인 또는 반대 중 하나를 선택하라는 메시지가 표시됩니다. 선택 후 확인하면, 해당 투표는 다음 블록에 포함되어 집계됩니다.

```
btcli sudo senate-vote
```

다이내믹 TAO

다이내믹 타오(Dynamic TAO)는 비트센서의 통합 토크노믹스 및 거버넌스 모델의 주요 진화 형태로, 네트워크가 보다 완전한 탈중앙화(decentralization)를 향해 나아가는 중요한 전환점입니다.

이 모델은 제안서를 통해 제출되어 상원(Senate)의 투표로 승인되었으며, 약 1년간의 연구·개발 및 테스트 이후 2025년 2월 13일, 비트센서 메인 네트워크 업그레이드로 구현되었습니다.

합의 구조에 대한 점진적 영향

Dynamic TAO의 도입은 점진적으로 네트워크 합의 구조(consensus dynamics)에 영향을 미치도록 설계되어 있습니다. Dynamic TAO가 처음 배포될 때는, 모든 검증자(validator)의 스테이크 기반 가중치(weight)는 변하지 않습니다. 그 이유는 TAO와 알파(α) 화폐 간의 상대적 가중치를 제어하는 TAO 가중치(TAO weight)라는 편향 변수(biasing variable)가 존재하기 때문입니다. 초기 단계에서는 이 TAO 가중치가 TAO에 절대적으로 유리하게 설정됩니다. 즉, 현재 알파 화폐(alpha currencies)가 존재하지 않기 때문에 TAO의 가중치는 100%로 유지됩니다. 시간이 지나면서 (약 100일이 예상됨), 이 TAO 가중치는 점진적으로 조정되어 알파 화폐 쪽으로 가중치가 이동(favor alpha currencies)하게 됩니다. 이로써 TAO 기반의 스테이크 중심 구조에서 알파 기반의 서브넷 경제로 서서히 권한과 보상의 균형이 이동하도록 설계되어 있습니다.

FAQ

일반 사항

알파(Alpha) 통화에 상한(cap)이 있나요?

네. 각 서브넷의 알파 토큰에는 2,100만 개의 하드 캡(hard cap)이 존재하며, 이는 TAO와 동일합니다. 알파 토큰 또한 TAO와 동일하게 반감기(halving schedule)를 따릅니다.

현재 기준으로, 각 서브넷당 블록마다 2개의 알파 토큰이 발행되며, 전체 네트워크에서 공유되는 1개의 TAO가 동시에 발행됩니다.

TAO 보유자 / 스테이커

스테이킹 방식은 어떻게 달라졌나요?

기존에는 검증자에게 직접 TAO를 스테이킹했지만, 다이내믹 타오에서는 특정 서브넷 내의 검증자에게 스테이킹을 수행합니다.

이때의 서브넷은 마이닝 서브넷(mining subnet)(대부분의 서브넷)일 수도 있고, 유일한 루트 서브넷(root subnet), Subnet Zero일 수도 있습니다.

- **마이닝 서브넷에서 스테이킹할 경우:** TAO를 해당 서브넷의 다이내믹 토큰인 알파(alpha)로 교환하고, 그 알파를 검증자의 핫키(hotkey)에 스테이킹합니다.
- **루트 서브넷에서 스테이킹할 경우:** TAO를 그대로 스테이킹하며, 발행 보상(emission)은 TAO로 지급됩니다.

서브넷에 스테이킹할 때의 위험/보상 구조는 어떻게 되나요?

각 신규 서브넷은 고유한 토큰인 알파(alpha)를 가집니다. 사용자가 특정 서브넷의 검증자에게 스테이킹할 때는 TAO를 해당 서브넷의 알파로 교환하게 됩니다. 반대로, 언스테이킹 시에는 알파를 다시 TAO로 교환합니다. 따라서 스테이킹 및 언스테이킹은 알파의 가격에 민감하게 반응합니다. 서브넷의 알파 가격은 서브넷 내 TAO 준비금 대비 알파 준비금의 비율로 결정됩니다.

즉, 서브넷에 TAO를 스테이킹한다는 것은 TAO를 그 서브넷의 알파 토큰으로 교환하는 행위이며, 이를 청산(exit)하려면 알파를 다시 TAO로 환전해야 합니다.

보유 중인 스테이크(즉, 알파 토큰)는 알파의 가격 변동에 따라 TAO 기준 가치가 증가하거나 감소할 수 있습니다.

루트 서브넷(Subnet 0) 스테이커의 발행 보상은 어떻게 작동하나요?

네트워크 전체 영향 : 루트 서브넷 스테이커의 스테이크는 검증자가 활동하는 모든 서브넷에 걸쳐 가중치(weight)를 제공합니다. 즉, 하나의 스테이크가 여러 서브넷에서 동시에 발행 보상을 추출하게 됩니다.

비례적 발행 및 TAO 가중치:

TAO와 알파는 검증자의 각 토큰 보유량 비율에 따라 스테이커에게 비례적으로 분배됩니다.

사용자가 알파 토큰(서브넷 토큰)을 전송할 수 있나요?

이는 서브넷 생성자에게 달려 있으며, TransferToggle 하이퍼파라미터를 통해 설정됩니다. 활성화된 경우, 알파 스테이크 보유자는 `btcli stake transfer` 또는 `transfer_stake` 명령어를 사용하여 해당 소유권을 다른 콜드키(coldkey)/지갑으로 이전할 수 있습니다.

Dynamic TAO는 네트워크 거버넌스에 어떤 영향을 미치나요?

Dynamic TAO는 비트텐서의 온체인 거버넌스 메커니즘(즉, 제안과 투표)에 직접적인 변화를 주지 않습니다.

서브넷

서브넷 제로

서브넷 제로(Subnet Zero), 즉 루트 서브넷(root subnet)은 특별한 서브넷입니다. 이 서브넷에는 채굴자가 등록할 수 없으며, 검증 작업도 수행되지 않습니다. 그러나 검증자는 등록할 수 있으며, τ 보유자는 다른 서브넷과 마찬가지로 해당 검증자에게 스테이킹할 수 있습니다. 이 구조는 τ 보유자가 특정 서브넷에 구애받지 않고(subnet-agnostic way) 검증자에게 τ 를 스테이킹할 수 있는 메커니즘을 제공합니다. 이는 서브넷 내 검증자의 가중치가 그 서브넷의 α 지분뿐 아니라, 서브넷 제로에 스테이킹된 TAO의 비율도 포함하기 때문에 가능합니다.

서브넷 제로(Subnet Zero)는 종종 루트 서브넷이라고 불립니다. 그 이유는, Dynamic TAO 도입 이전 구조에서 루트 네트워크(root network)가 담당하던 역할을 어느 정도 대체하기 때문입니다. 그러나 서브넷 제로는 루트 네트워크의 핵심 기능이었던 서브넷 간 합의를 수행하지 않습니다. 즉, 이름은 유사하지만 실제로는 루트 네트워크의 합의 기능을 계승하지 않으며, 주로 TAO 스테이킹 및 검증자 등록과 같은 관리적 기능만 담당합니다.

Dynamic TAO에서 서브넷을 시작하고 관리하기 위해 필요한 것

Dynamic TAO에서 서브넷을 등록하는 과정은 이전 방식과 매우 유사하지만, 이제 서브넷 등록 비용은 더 이상 잠금 후 해제되는(lock-and-return) 형태가 아니라 소각(burned)됩니다. 이는 Dynamic TAO에서는 서브넷 등록이 해제(deregistered)되지 않기 때문입니다.

서브넷 등록 비용

서브넷 등록 비용은 동적(dynamic)으로 조정됩니다. 새로운 서브넷이 등록될 때마다 비용이 두 배로 증가하며, 이후 점진적으로 감소하여 약 38,880 불록(약 5.5일)이 지나면 절반으로 줄어듭니다. 따라서 새로운 서브넷에 대한 수요가 일정하다면, 약 5일 반마다 하나의 서브넷이 생성되는 것이 이상적입니다.

Dynamic TAO가 서브넷 거버넌스(가중치 설정)에 미치는 영향

각 검증자의 서브넷 내 가중치(weight)는 그 서브넷에서 해당 검증자에게 스테이킹된 알파(α)의 양과 서브넷 제로(Subnet Zero)에서 그에게 스테이킹된 TAO의 양의 함수로 결정된다. 이때 TAO의 가치는 TAO 가중치(TAO weight)로 불리는 값에 의해 조정되며, 이 가중치는 0과 1 사이의 값을 가진다.

서브넷이 방치되면 어떻게 되는가?

만약 어떤 서브넷이 사용되지 않거나 채굴이 중단된다면, 해당 서브넷의 토큰은 거의 무가치한 수준으로 하락할 가능성이 높습니다. 다른 서브넷이나 루트 네트워크(root)는 이에 영향을 받지 않습니다. 각 서브넷의 성공과 실패는 대체로 독립적으로 이루어집니다.

현재 프로토콜은 서브넷을 자동으로 해제(deregister)하지 않습니다. 방치된 서브넷(abandoned subnet)은 다시 활성화(revive)될 수 있습니다.

서브넷 생성자는 자신이 만든 토큰의 발행량을 통제할 수 있는가?

아닙니다. 발행량(emission)은 프로토콜 로직(예: `run_coinbase.rs`)에 의해 계산되며, 네트워크 전체 파라미터를 기반으로 합니다. 즉, 서브넷 창립자는 임의로 토큰을 발행(print)할 수 없으며, 발행량은 모든 서브넷에서 동일한 일관된 규칙에 따라 결정됩니다.

Dynamic TAO 도입 이전 서브넷의 기존 등록 비용은 어떻게 되는가?

이전 서브넷에서 잠금(lock)되어 있던 등록 비용은 Dynamic TAO가 시작될 때, 해당 서브넷을 등록했던 동일한 콜드키(coldkey)로 반환됩니다.

채굴자와 검증자

채굴자와 검증자는 서브넷의 급증(proliferation)을 어떻게 관리할 수 있는가?

채굴자와 검증자는 이제 자신이 채굴하는 알파 토큰(alpha token)의 TAO 가치를 고려해야 합니다.

각 서브넷의 가치와 자신이 경쟁할 수 있는 능력에 따라, 채굴자와 검증자는 작업을 서브넷 간에 이동해야 하며, 서브넷의 수가 많아질수록 보다 전문화된 작업 기회도 늘어납니다.

채굴자와 검증자는 최적의 자원 할당(allocation)을 위해 시장 상황(토큰 가격, 거래량 등)을 모니터링할 필요가 있습니다.

이는 다른 작업증명(Proof of Work, PoW) 시스템에서 채굴자들이 어떤 체인을 채굴하는 것이 가장 수익성 높은지를 판단하기 위해 시장을 관찰하는 것과 유사합니다.

비트텐서 커뮤니티 링크

비트텐서 생태계에 오신 것을 환영합니다!

이 페이지는 비트텐서 네트워크를 탐색하는 데 도움이 되는 필수 커뮤니티 도구 및 자료에 대한 링크를 제공합니다.

또한 [서브넷 목록](#)을 통해 비트텐서의 다양한 서브넷을 탐색하고, 각 서브넷의 웹사이트 및 저장소(repository) 링크를 확인할 수 있습니다.

Opentensor 재단

- **트위터:** [@opentensor](#) - 공식 공지 및 업데이트
- **팟캐스트:** [Novelty Search](#) - Opentensor Foundation 공식 팟캐스트
- **디스코드:** [비트텐서 Discord 서버 참여하기](#) - 커뮤니티와 소통하고, 질문하며, 최신 소식을 받아보세요.

블록 익스플로러

- [TAO.app](#) - 추천 (이 문서와 비트텐서 핵심 생태계 기여를 담당하는 *Latent* 팀에서 관리). *Savant*, 체인 탐색 AI 어시스턴트가 포함되어 있습니다.
- [Taostats](#)
- [Taomarketcap](#)

지갑 애플리케이션

- [비트텐서 Wallet](#) - 추천 (OTF에서 관리).
- [기타 Polkadot 지갑들](#) - 비트텐서와 상호작용할 수 있는 멀티체인 지갑.

개발 리소스

- [BTCLI](#) - 비트텐서 명령줄 인터페이스(CLI)
- [비트텐서 SDK](#)
- [Subtensor](#) - 비트텐서의 Substrate 기반 블록체인

토크노믹스

TAO는 비트텐서 네트워크의 참여자들에게 보상을 제공하는 유틸리티 토큰입니다. 여기에는 [채굴자\(miners\)](#), [검증자\(validators\)](#), 그리고 [서브넷 소유자\(subnet owners\)](#)가 포함됩니다. 채굴자들은 AI 작업을 위한 필수적인 연산 자원을 제공합니다. 검증자들은 [합의\(consensus\)](#) 규칙이 올바르게 유지되도록 보장합니다. 이로써 네트워크의 무결성이 유지됩니다.

RAO란?

RAO는 TAO의 단위로, TAO의 10억분의 1 ($(10^{\{-9\}})$)을 나타냅니다.

TAO 유틸리티

TAO 토큰은 비트텐서 생태계 내에서 여러 중요한 역할을 수행합니다.

- **인센티브 제공:** TAO는 사용자가 가치 있는 자원을 제공하도록 유도합니다. 이러한 자원에는 컴퓨팅 파워와 데이터가 포함됩니다. 참여자는 자신의 기여에 대한 보상으로 TAO 토큰을 획득합니다. 이는 더 많은 개인이 네트워크에 참여하도록 동기를 부여합니다.
- **스테이킹(Staking):** 사용자는 TAO 토큰을 스테이킹하여 합의 메커니즘(consensus mechanism)에 참여할 수 있습니다. 스테이킹은 일종의 담보(collateral) 역할을 하며, 검증자가 네트워크의 안정성과 건전성을 우선시하도록 만듭니다. 또한 악의적인 행위자에 대한 보안성을 강화합니다.
- **거래 수수료(Transaction Fees):** 비트텐서 네트워크 내에서 발생하는 모든 거래는 TAO 토큰으로 수수료를 지불해야 합니다.

비트코인과의 비교

비트텐서는 비트코인과 몇 가지 핵심적인 유사점을 공유하지만, 그 초점은 통화가 아닌 탈중앙화된 인공지능(Decentralized AI)에 있습니다.

목적	토크노믹스	합의 메커니즘
비트코인	탈중앙화 통화	고정 공급량 및 반감기 작업 증명(Proof-Of-Work)
비트텐서	탈중앙화 인공지능	고정 공급량 및 반감기 지능 증명(Proof-Of-Intelligence)

이 두 네트워크는 다음과 같은 주요 공통점을 가지고 있습니다.

- **목적:** 비트코인은 은행 없이 돈을 관리할 수 있는 새로운 방식을 창조했습니다. 이와 유사하게, 비트텐서는 대형 기술 기업(Big Tech)의 통제를 받지 않고 인공지능을 구축할 수 있는 새로운 방식을 제시합니다. 이를 통해 전 세계 사람들이 자유롭게 협력할 수 있습니다.
- **토크노믹스:** 비트코인과 비트텐서 모두 총 발행량이 2,100만 개로 고정되어 있습니다. 두 네트워크는 유사한 반감기(halving) 일정을 통해 시

간이 지남에 따라 토큰의 발행 속도를 조절합니다. 반감기(Halving)는 약 4년마다 한 번씩 발생하며, 보상이 절반으로 줄어듭니다.

- **합의 메커니즘:** 비트코인은 복잡한 수학적 문제를 해결하기 위해 경쟁하는 작업 증명(Proof-of-Work) 방식을 사용합니다. 이 과정을 통해 네트워크가 보호되고, 각 문제를 해결할 때마다 거래가 검증되며 보상이 지급됩니다. 반면, 비트덴서는 지능 증명(Proof-of-Intelligence) 접근법을 채택합니다. 참여자는 유용한 AI 모델을 네트워크에 기여함으로써 과제를 해결하고 보상을 받습니다. 두 시스템 모두 네트워크를 강화하는 사람들에게 공정한 보상을 제공하는 메커니즘을 형성합니다.

경제 모델(Economic Model)

앞서 언급했듯이, TAO의 총 공급량은 2,100만 개로 제한되어 있으며 예측 가능한 속도로 생성됩니다. 각 블록은 1 TAO를 생성하며, 하루에 약 7,200개 블록이 생산됩니다. 각 블록은 약 12초마다 완성됩니다. 발행률은 약 4년마다 한 번씩 반감기를 거치며, 모든 토큰이 유통될 때까지 지속됩니다. TAO의 수요가 증가함에 따라, 제한된 공급으로 인해 그 가치가 상승할 가능성이 있습니다.

가치 제안

TAO의 가치는 채굴자와 검증자의 기여에 의해 결정됩니다. 네트워크에 더 많은 사용자가 참여하고 가치 있는 AI 서비스를 제공할수록 TAO의 수요는 증가합니다. 이로 인해 TAO의 생태계 내 가치는 더욱 강화됩니다.

발행 보상 (Emissions)

발행 보상(Emission)은 새롭게 생성된 토큰으로, 각 블록(block)마다 (약 12초마다) 다음에 분배됩니다.

- 서브넷 풀(Subnet pools): TAO 및 ALPHA
- 서브넷 참여자(Subnet participants): ALPHA

이중 투입(injection) 메커니즘은 경제적 균형을 유지하면서 기여자에게 보상을 제공합니다.

서브넷 토큰을 ALPHA라고 부르는 이유

각 서브넷은 고유한 토큰을 보유합니다 (예: 서브넷 1은 ALPHA, 서브넷 2는 BETA 등). 편의상 모든 서브넷 토큰을 “ALPHA”라고 통칭합니다.

TAO 발행 보상(Emissions)의 작동 방식

각 서브넷이 TAO 블록 보상에서 차지하는 비율은 해당 서브넷의 EMA 가격이 다른 모든 서브넷의 EMA 가격과 비교하여 얼마나 높은지에 따라 결정됩니다. 다른 서브넷들에 비해 특정 서브넷의 EMA 가격이 높을수록, 그 서브넷이 받는 발행 보상(emission) 비율도 커집니다. TAO의 블록 보상은 현재의 총 공급량에 따라 달라집니다. 반감기(halving) 이전에는 TAO의 블록 보상은 1입니다.

TAO 보상 계산의 기본 공식은 다음과 같습니다:

$$\text{TAO}_{\text{emission}} = \left(\frac{\text{EMA}_{\text{subnet}}}{\text{EMA}_{\text{sum}}} \right) \times \text{Block Emission}$$

예시 (모든 EMA 가격의 합 = 1.0, 블록 보상 1 TAO로 가정):

- 서브넷 1은 $(0.5 \div 1.0) \times 1 = 0.50$ TAO
- 서브넷 2는 $(0.2 \div 1.0) \times 1 = 0.20$ TAO
- 서브넷 3은 $(0.3 \div 1.0) \times 1 = 0.30$ TAO

루트 서브넷의 보상(Emissions on Root Subnet)

서브넷 0(Root)은 EMA 가격 합계 계산에서 제외되며, 보상(emission)에서도 제외됩니다.

ALPHA 발행 보상(Emissions)의 작동 방식

서브넷은 독립적으로 ALPHA 보상을 받으며, 각 서브넷당 최대 2 ALPHA가 블록당 생성됩니다. ALPHA의 블록 보상 기본값은 TAO와 동일하게 블록당 1이지만, ALPHA는 두 구성 요소 간의 분배를 위해 두 배로 발행됩니다.

- 풀 리저브(Pool Reserve, ALPHA in): 서브넷의 풀로 전달됩니다.
- 참여자(Participants, ALPHA out): 채굴자(miners), 검증자(validators), 서브넷 소유자(subnet owners)에게 분배됩니다.

풀로 유입되는 ALPHA의 양은 다음 공식으로 계산됩니다:

$$\text{ALPHA}_{in} = \frac{\text{TAO}_{\text{emission}}}{\text{ALPHA}_{\text{price}}}$$

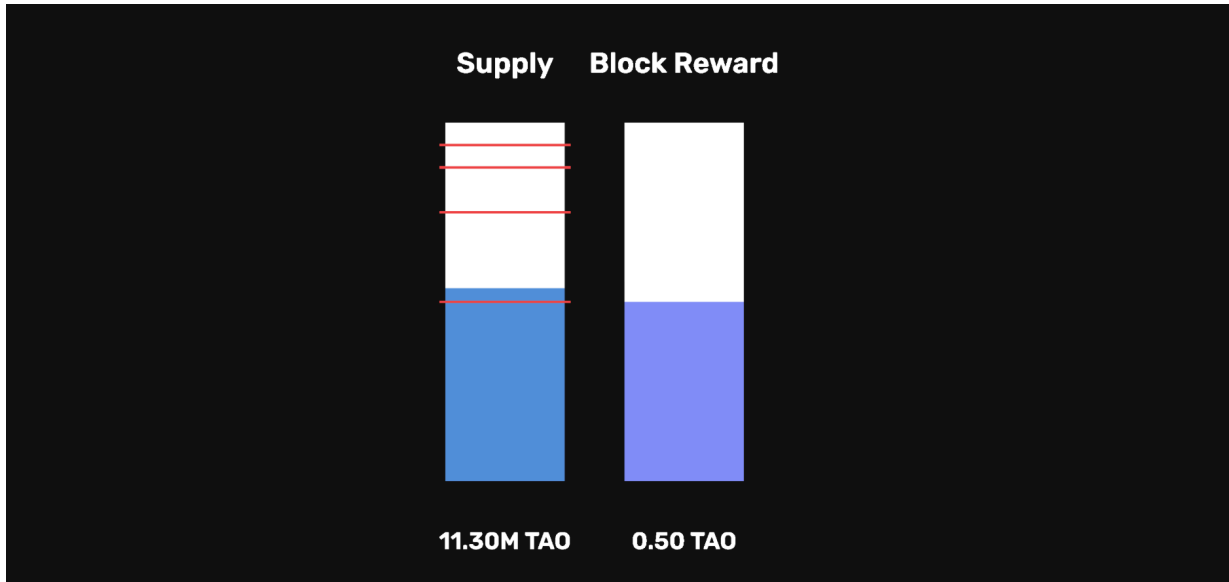
이 공식은 TAO 보상과 ALPHA 가격 간의 동적 균형을 형성합니다. ALPHA의 가격이 서브넷의 현재 TAO 보상보다 현저히 높을수록, 풀로 유입되는 ALPHA의 양은 줄어듭니다.

참여자에게 분배되는 ALPHA는 반대로 항상 ALPHA 블록 보상의 기준값으로 설정됩니다. 이 메커니즘은 풀로 유입되는 ALPHA가 적을 때 전체 ALPHA 인플레이션이 낮아지도록 보장합니다. 즉, 최대 2 ALPHA가 발행될 수 있지만 실제로는 그보다 적게 발행되어 인플레이션이 억제됩니다.

한 서브넷이 2 ALPHA를 전량 발행하는 유일한 경우는 해당 서브넷의 ALPHA 가격이 TAO 보상과 같거나 그보다 낮을 때입니다. 이 경우 ALPHA 보상은 자동으로 다운스케일링(downscaling)됩니다.

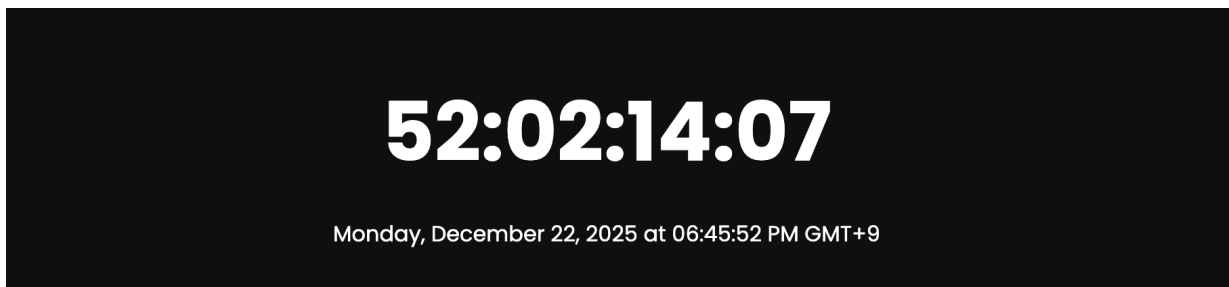
반감기(Halving)

비트텐서의 반감기(Halving)는 새로운 TAO 토큰이 유통에 들어오는 속도를 줄이는 역할을 합니다. 비트코인이 블록(block) 수를 기준으로 반감기를 적용하는 것과 달리, 비트텐서는 총 토큰 공급량(total supply)을 기준으로 반감기를 수행합니다. 특정한 공급 임계치에 도달하면 TAO의 보상(emission) 속도가 절반으로 줄어듭니다.



언제 발생하나요?

첫 번째 반감기(halving)는 예측컨대 하기 시점에 발생할 예정입니다:

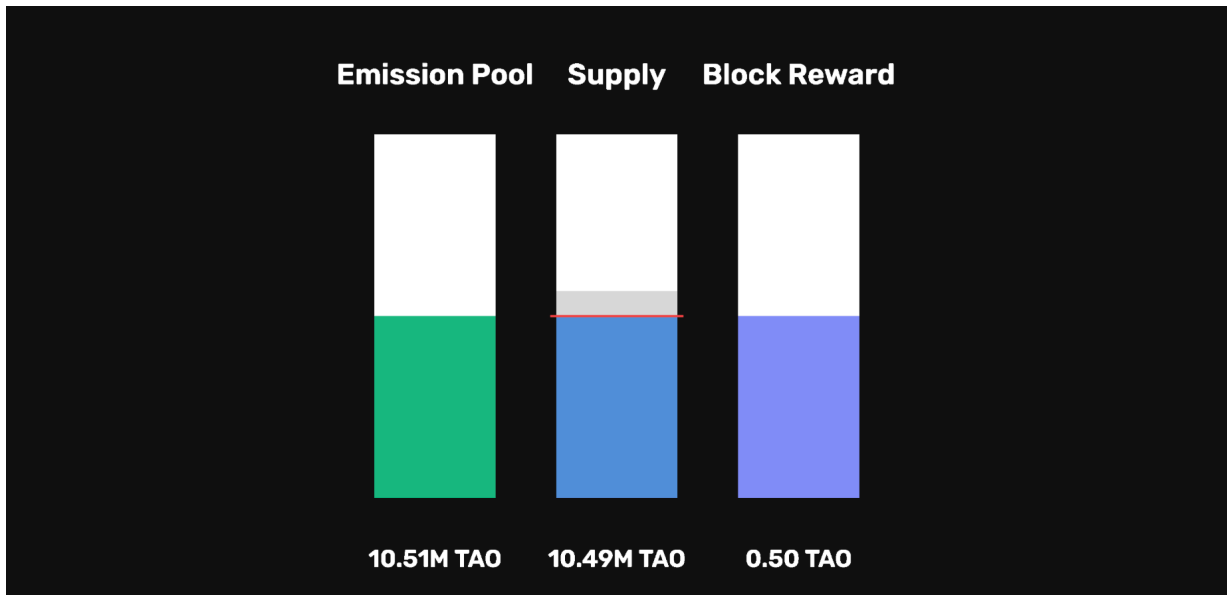


반감기 시점(Halving Date)

실제 반감기 시점은 매일 리사이클링되는 TAO의 양에 따라 지속적으로 변동됩니다. 위의 예측치는 현재의 리사이클링(recycling) 속도가 유지된다는 가정하에 계산된 것입니다.

TIP: 데이터는 실시간으로 갱신되므로, 주기적으로 타이머를 다시 확인하는 것이 좋습니다.

리사이클링 지연



리사이클링된 등록 수수료는 반감기 시점을 지연시킬 수 있습니다. 토큰이 리사이클링되면 이들은 다시 보상 풀로 돌아가며, 따라서 유통 공급량에서 제외됩니다. 이 과정은 다음 반감기 이벤트까지의 시간을 연장시키는 효과를 가집니다. 리사이클링되는 토큰이 많을수록, 다음 반감기에 도달하는 데 더 오랜 시간이 걸립니다.

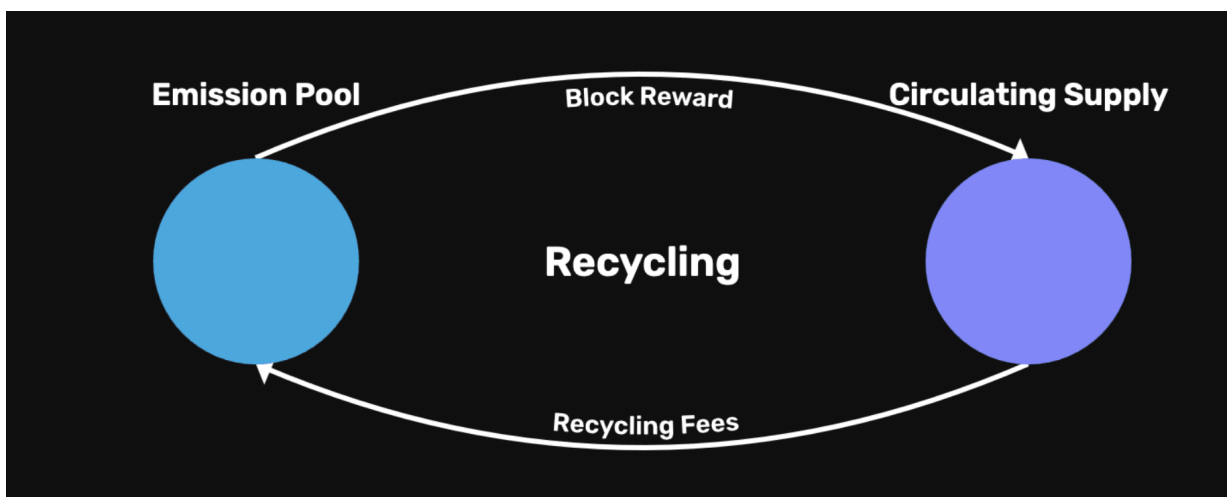
네트워크에 미치는 영향

반감기는 네트워크의 모든 참여자에게 영향을 미칩니다. 현재 보상(emission)은 하루에 7,200 TAO를 유통 공급량에 추가합니다 (리사이클링을 고려하지 않은 기준). 첫 번째 반감기 이후에는 이 속도가 하루 3,600 TAO로 감소합니다. 이 보상은 채굴자(miners), 검증자(validators), 서브넷 소유자(subnet owners)에게 분배됩니다. 모든 참여자는 동일한 작업에 대해 더 적은 보상을 받게 됩니다. 이러한 공급 감소는 토큰, 즉 TAO의 가치 상승 압력(value pressure)을 유발할 수 있습니다.

리사이클링(Recycling)

TAO 토큰을 비트텐서의 보상 시스템으로 되돌려 향후 분배에 재활용하는 과정입니다.

리사이클링은 마치 토큰을 다시 보상 풀에 넣는 것과 같습니다. 채굴자(miners)나 검증자(validators)가 서브넷에 등록할 때, 그들의 TAO 토큰은 사라지지 않습니다. 대신 이 토큰들은 네트워크의 보상 풀(emission pool)로 되돌아가며, 이를 통해 다음 반감기(halving) 시점까지의 기간이 연장됩니다.



리사이클링 유형

네트워크에서 TAO가 재활용되는 주요 활동은 두 가지입니다:

- 신규 채굴자 및 검증자의 등록 수수료(registration fees)
- 등록 기능이 비활성화된 서브넷에서 발생하는 보상(emissions)

루트 네트워크(Root Network)

루트(root) 서브넷은 항상 자신이 생성한 보상을 시스템으로 다시 재활용합니다. 등록 기능을 비활성화한 서브넷 역시 동일한 패턴을 따릅니다. 이러한 서브넷들은 토큰을 직접 분배하는 대신, TAO를 네트워크의 보상 풀(emission pool)로 되돌려 보냅니다.

스테이킹/위임

사용자가 자신의 TAO 토큰을 예치하여 검증자를 지원하고 보상을 얻는 과정입니다.

스테이킹(Staking)이란 네트워크를 지원하기 위해 TAO 토큰을 예치(lock-up)하는 것을 의미합니다. 스테이킹을 하는 사용자는 네트워크의 보안 강화에 기여하며, 예치한 금액에 따라 보상을 받습니다. 최소 스테이킹 금액은 0.1 TAO입니다.

위임(Delegation)은 사용자가 직접 노드를 운영하지 않고도 검증자(validators)를 지원할 수 있게 합니다. TAO를 위임하면, 사용자의 스테이킹 권한이 검증자에게 대여되어 검증자는 위임받은 자(delegatee)가 됩니다. 사용자는 토큰의 소유권을 계속 유지하며, 일정 빈도 제한 내에서 언제든지 스테이킹을 해제(언스테이킹)할 수 있습니다.

스테이킹 관련 정보

TAO를 스테이킹하거나 언스테이킹할 때 별도의 락업(lock-up) 기간은 없습니다.

사용자는 필요에 따라 자유롭게 스테이킹 비율을 조정할 수 있습니다.

자신의 TAO는 항상 본인의 통제하에 있으며, 검증자가 직접적으로 자금을 접근하거나 이동시킬 수 없습니다.

스테이킹 보상은 taostats.io와 같은 도구를 통해 실시간으로 확인할 수 있습니다.

작동 방식

사용자가 TAO를 검증자에게 위임하면, 그 검증자의 유효 스테이크(effective stake)가 증가합니다. 스테이크가 클수록 검증자가 가중치(weights)를 설정하는 과정에서 더 큰 영향력을 갖게 됩니다. 검증자는 자신의 스테이크와 위임받은 스테이크를 모두 모아 검증(validation) 작업을 수행합니다.

검증 작업으로부터 발생한 보상은 모든 이해관계자(stakeholders)에게 분배됩니다. 검증자는 일반적으로 보상의 0-18%를 유지하고, 나머지는 자신의 스테이킹 비율에 따라 후보자(nominators) 또는 위임자(delegators)에게 분배합니다. 이 보상은 네트워크에서 매일 분배됩니다.

보상 분배

네트워크는 총 스테이크와 성과(performance)에 따라 보상을 계산합니다. 검증자의 자체 스테이크는 전체 보상을 받으며, 위임된 스테이크의 보상은 위임받은 자(delegatee)와 위임자들(nominators) 간에 분배됩니다. 검증자가 가져가는 비율은 0%에서 18% 사이이며, 이는 각 검증자가 직접 설정합니다.

예시:

한 검증자가 900 TAO를 보유하고 있으며, 세 명의 후보자(nominators)가 다음과 같이 추가로 위임했습니다:

- 후보자 1: 120 TAO 위임
- 후보자 2: 50 TAO 위임
- 후보자 3: 30 TAO 위임

이로써 총 유효 스테이크는 1,100 TAO가 됩니다 ($900 + 120 + 50 + 30$).

위임된 200 TAO 중 각 후보자의 비중은 다음과 같습니다:

- 후보자 1: $120/200 = 0.6$ (60%)
- 후보자 2: $50/200 = 0.25$ (25%)
- 후보자 3: $30/200 = 0.15$ (15%)

이제 검증자가 100 TAO의 보상을 획득했다고 가정합니다.

검증자는 자신의 스테이크 비율에 따라 우선 몫을 가져갑니다.

- 검증자의 기본 몫: $100 \times (900 / 1100) = 81.82$ TAO

검증자는 위임된 보상의 9%를 추가로 가져갑니다. 위임된 보상은 다음과 같습니다:

- 위임 보상: $100 \times (200 / 1100) = 18.18$ TAO

따라서 검증자는 $18.18 \times 0.09 = 1.64$ TAO를 추가로 획득합니다.

총합으로 검증자는 83.46 TAO ($81.82 + 1.64$)를 받게 됩니다.

남은 16.54 TAO ($100 - 83.46$)는 후보자들에게 그들의 기여 비율에 따라 분배됩니다:

- 후보자 1: 9.92 TAO ($16.54 \times 60\%$)
- 후보자 2: 4.14 TAO ($16.54 \times 25\%$)
- 후보자 3: 2.48 TAO ($16.54 \times 15\%$)

루트 클레임(Root Claim)

루트 클레임은 사용자가 루트 서브넷(root subnet)에 스테이킹할 때 배당(dividend)을 어떻게 수령할지를 결정할 수 있게 해주는 기능입니다. 밸리 데이터(valuator)는 자신이 검증하는 모든 서브넷에서 ALPHA를 획득하며, 사용자는 그 배당금의 비율에 따라 지분을 받게 됩니다. 루트 클레임은 이러한 ALPHA를 원래 형태로 유지할지, 아니면 자동으로 TAO로 변환할지를 결정할 수 있게 해줍니다.

루트 클레임이란?

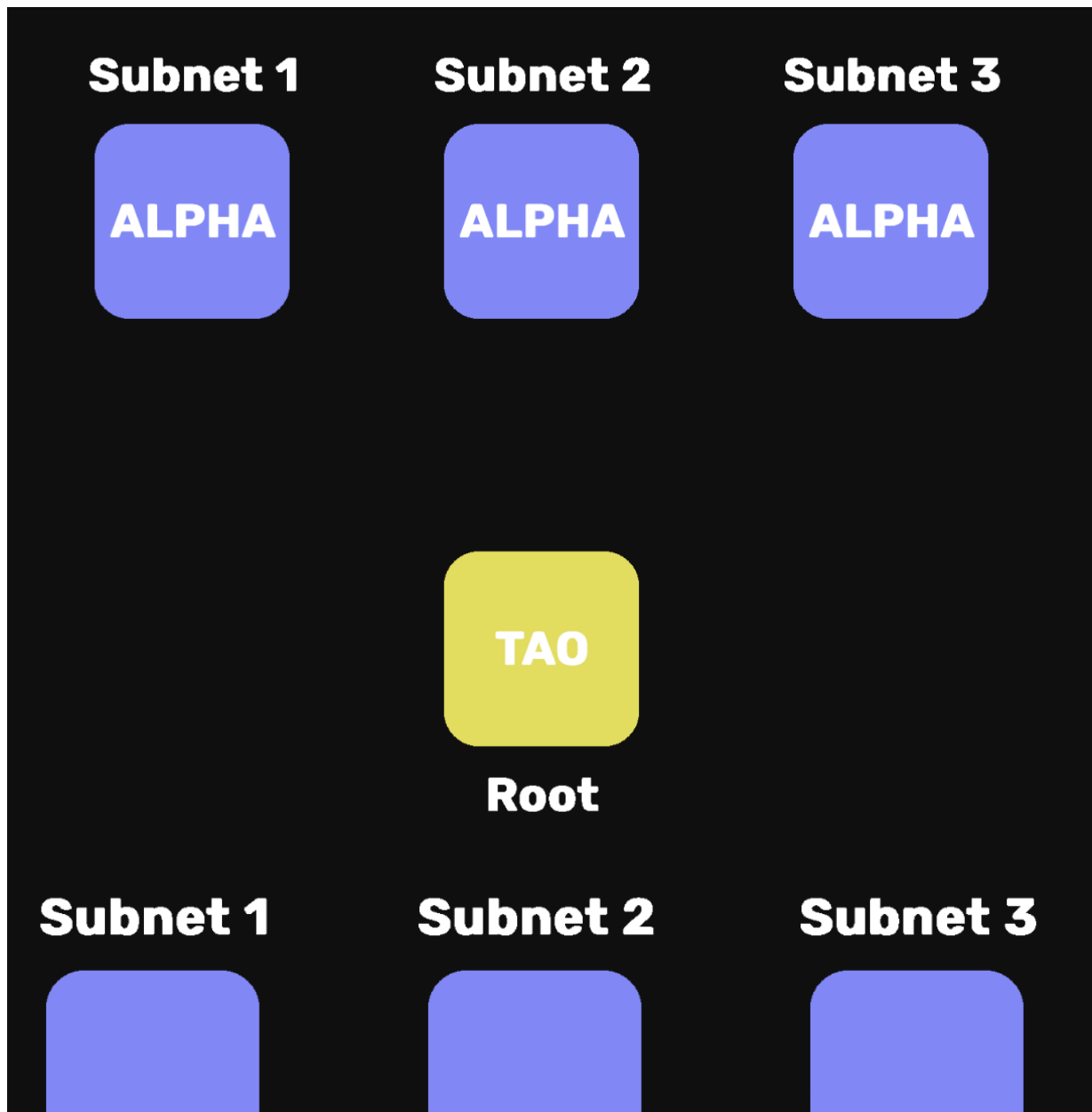
사용자가 밸리데이터를 통해 루트 서브넷에 TAO를 스테이킹하면, 해당 밸리데이터가 검증하는 모든 서브넷으로부터 ALPHA 배당을 받을 자격이 생깁니다. 이러한 배당은 자동으로 누적됩니다. 루트 클레임은 사용자가 배당을 청구(claim)할 때 그 배당금이 어떻게 처리될지를 선택할 수 있게 해주는 기능입니다.

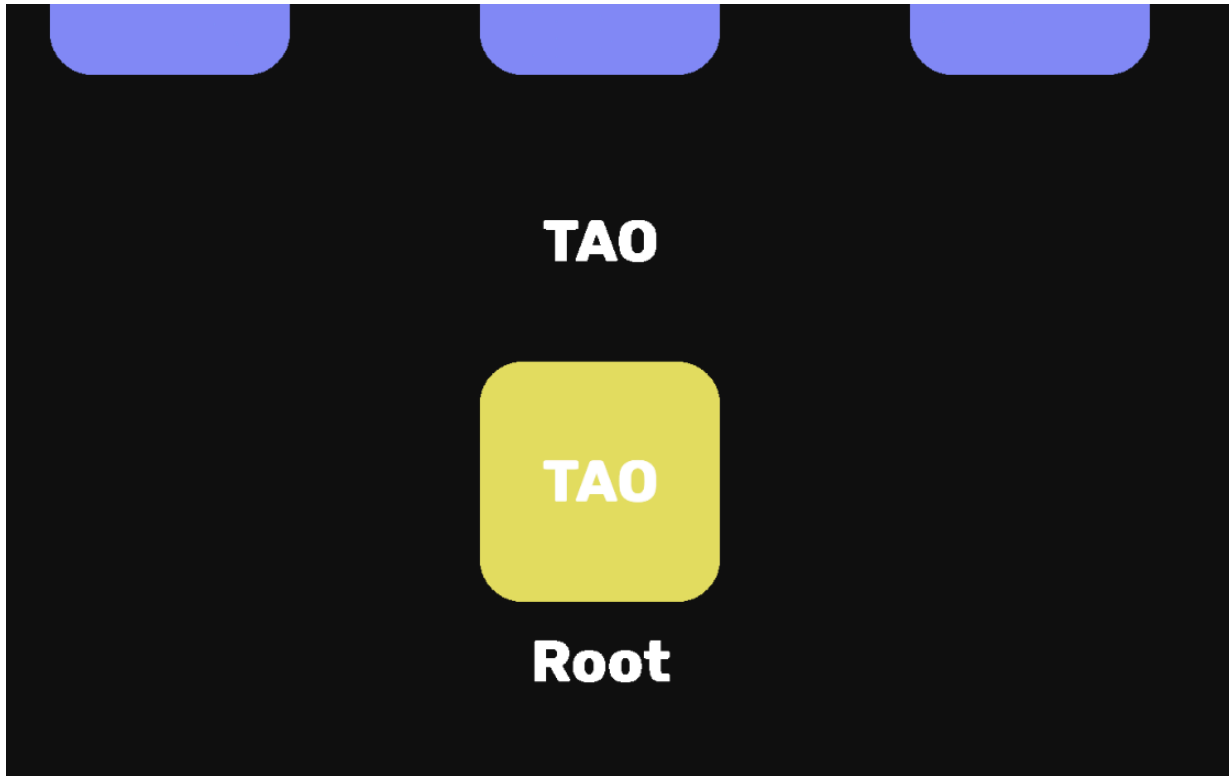
배당 청구(CLAIMING OF DIVIDENDS)

이 메커니즘은 TAO를 루트 서브넷에 스테이킹한 계정에만 적용됩니다. TAO를 직접 서브넷에 스테이킹한 경우에는 ALPHA 배당이 즉시 지급되며, 루트 외부 스테이킹 계정에는 청구 절차가 존재하지 않습니다.

스왑(Swap) 클레임

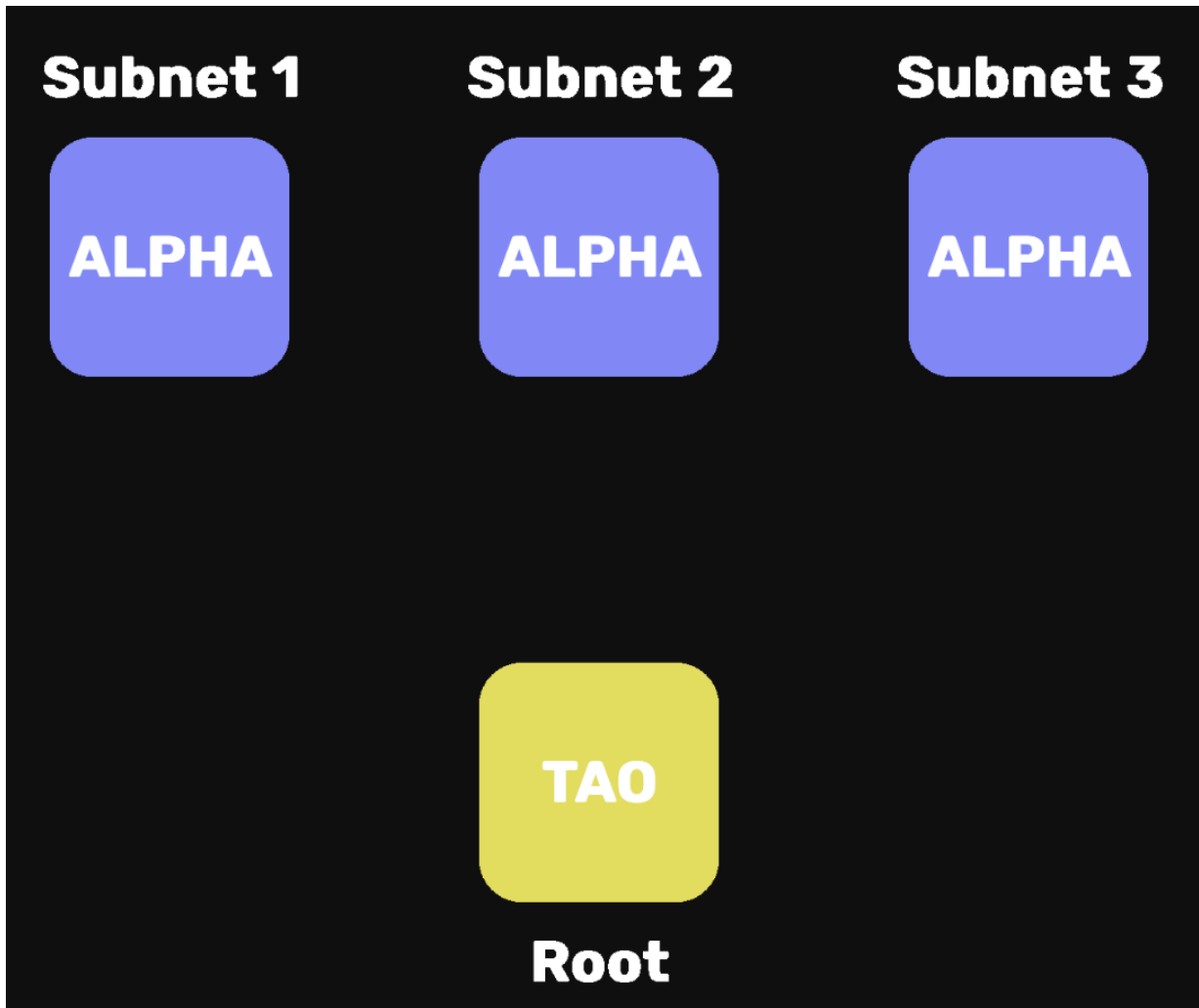
사용자는 루트에서 누적된 배당금을 클레임하는 두 가지 방식 중 하나를 선택할 수 있습니다. “스왑(Swap)” 클레임은 누적된 ALPHA를 현재 ALPHA 가격에 따라 TAO로 변환한 뒤, 자동으로 그 TAO를 루트에 다시 스테이킹합니다. 이는 기본 옵션이며, 루트 클레임이 도입되기 전의 기존 동작 방식을 반영합니다.





킵(Keep) 클레임

킵(Keep) 클레임은 ALPHA를 ALPHA가 생성된 서브넷에 직접 스테이킹합니다. ALPHA의 가격이 상승하면 해당 서브넷의 스테이킹 가치가 더 많은 TAO에 해당하게 되고, ALPHA의 가격이 하락하면 가치가 줄어듭니다. 이 방식은 루트 서브넷에 스테이킹된 초기 TAO를 그대로 유지하면서, ALPHA 토큰에 대한 서브넷 익스포저를 얻을 수 있게 해줍니다.



TAO 토큰은 루트에 계속 스테이킹된 상태로 유지되며, 서버넷에서 ALPHA 스테이킹이 점차 증가하게 됩니다. ALPHA 토큰을 서버넷에 유지하도록 선택하면, 이를 TAO로 변환하는 기본 동작으로 인해 발생하는 매도 압력(sell pressure)을 완화하는 효과가 있으며, 이는 ALPHA 가격 하락을 억제하는 데 기여합니다.

자동 및 수동 클레임

블록체인은 회전 방식으로 계정을 선택하여 각 블록마다 클레임을 처리합니다. 블록체인은 블록당 최대 5개의 서로 다른 계정만 처리할 수 있습니다. 스테이킹 계정의 전체 수에 따라 클레임 빈도는 달라질 수 있습니다. 작성 시점 기준으로, 자동으로 클레임이 실행되기까지 최대 2일이 걸릴 수 있습니다.

기다릴 수 없는 사용자는 수동으로 클레임할 수 있습니다. 사용자는 최대 5개의 서버넷을 선택하여 배당금을 청구할 수 있습니다. 사용자의 클레임 유형 선호도 — 스왑(Swap) 또는 킵(Keep) — 는 자동 또는 수동 클레임 모두에 동일하게 적용됩니다.

최소 임계값(MINIMUM THRESHOLD)

자동 클레임은 최소 0.0005 ALPHA의 임계값을 가집니다. 누적된 배당금이 이 금액보다 적으면 자동 클레임 대상이 되지 않습니다. 이는 작은 거래가 블록체인을 과도하게 부풀리는 것을 방지하기 위한 것입니다. 그러나 수동으로 클레임하는 경우에는 이 임계값 이하의 ALPHA 배당도 청구할 수 있습니다.