**[인공신경망(ANN)]**

인공신경망은 입력, 출력, 그리고 그 사이에 하나 이상의 은닉 계층을 포함하는 노드들로 구성되어 있습니다. 각 노드는 다른 노드에 연결되고, 노드의 출력이 지정된 임계값을 초과하면 해당 노드가 활성화되어, 다음 계층으로 정보를 전송합니다. 이런 신경망들 중 은닉 계층이 2개 이상인 신경망을 딥(deep) 하다고 하여 딥러닝이라고 부릅니다.

**[SNN]**

Spking Neural Network는 인공신경망의 한 종류로, 기존 딥러닝 네트워크들이 연속적인 값을 주고받는 것에 비해, binary 스파이크 신호를 기반으로 정보를 처리하는 인공신경망입니다. SNN은 실제 두뇌의 정보 전송 매커니즘을 모방한 스파이크 시퀀스를 활용해 모델의 뉴런들이 소통하는데, 이 뉴런들이 이전 계층의 입력을 막전위(membrane potential)라고 하는 내부 상태로 통합합니다.

SNN에서 많이 사용하는 뉴런 모델은 Integrate-and-Fire라고 하는 IF와, Leaky Integrate-and-Fire라고 하는 LIF가 있습니다. IF는 입력을 받을 때 마다 막전위의 값을 증가시켜 막전위가 임계값을 넘게 되면 스파이크를 발화하는 뉴런 모델이고, LIF는 IF 모델에 누수를 추가해 시간이 지날수록 막전위의 값이 줄어드는 뉴런 모델입니다.

**이것은 뉴런이 스파이크를 발화하는 방식을 나타낸 그림입니다.** pre-neuron으로부터 세개의 스파이크를 받아 post-neuron의 막전위가 특정 임계값을 넘게 되면 스파이크를 발화하고, 스파이크의 발화 여부에 따라 0과 1의 바이너리 형태(binary spike train)로 정보가 전달됩니다. 스파이크가 발화하면 막전위는 다시 초기화 되는데, 이런 초기화 방법에는 Hard와 Soft 두 종류가 있습니다. Hard는, 0 혹은 지정한 값으로 초기화 하는 것이고, Soft는 현재 값에서 스파이크의 임계값만큼 빼는 것입니다. SNN의 장점은 기존 DNN보다 낮은 전력을 소모한다는 것이며, 문제점은 아직 SNN에 최적인 학습 알고리즘 많이 개발되지 않아서, 대부분 기존 DNN에서 사용하는 알고리즘을 가져와 사용하고 있다는 것입니다.

**[NAS]**

현재 최고의 성능을 보이는 많은 DNN 아키텍처는 전문가들이 직접 디자인한 결과이지만, 특정 task와 dataset이 주어졌을 때, 어떠한 구조를 사용해야 하는지 일일이 정하는 일은 매우 어렵고 비효율적입니다. 모델을 구성하는 여러가지 매개변수들은, 모델을 구현하는 연구자가 직접 설정하고 학습을 시켜봐야 그 성능을 알 수 있고, 그렇게 만들어 놓은 모델은 task와 dataset이 바뀌었을 때, 똑같이 최적의 모델이라고 보장할 수 없기 때문입니다. Neural Architecture Search는 바로 이런 번거로운 작업을 자동화하여, 주어진 task에 가장 최적인 네트워크 구조를 편리하고 빠르게 탐색하는 방법을 연구하는 분야입니다. NAS는 **오른쪽 이미지와** 같은 3가지 단계로 진행되는데,

**가장 먼저**, 사용가능한 모든 operation의 집합인 후보 operation 세트와, 그 operation들이 어떻게 연결되고, 총 몇 개가 사용되는지에 대한 정의를 하게 됩니다. 그러므로 이 search space의 원소 조합에 따라 서로 다른 모델이 만들어지게 됩니다. **두 번쨰로**, 정의된 search space에서 지정된 탐색 전략에 따라 최적의 아키텍처를 탐색합니다. 본 논문에서는 이런 탐색 전략 중, random, greedy, bayesian이라는 세 가지 알고리즘을 사용했습니다. **세 번째로**, 탐색한 아키텍처에 대한 성능을 평가합니다. 이 때 본 논문에서 기준으로 하는 성능은 검증 정확도이며, dataset에서 학습과 검증을 8대2의 비율로 나눠 평가를 진행했습니다. 이렇게 평가된 성능을 기반으로 탐색 전략에 따라 특정 횟수만큼 탐색을 시도하게 됩니다.

**[포스터]**

높은 성능의 깊은 스파이킹 신경망 구조를 탐색하기 위해, DNN의 Neural Architecture 탐색 기술을 활용하는 연구들이 발표되고 있지만, 여러 탐색 알고리즘에 따른 NAS 성능에 대한 분석은 제시하고 있지 않았습니다. 따라서 저희는 SNN을 이용하여 3가지 NAS 알고리즘의 성능을 분석한 논문을 작성했습니다. 본 논문에서는 method의 오른쪽 그림과 같이 convolution block 5개, dense block 3개를 사용해 아키텍처를 탐색했고, 각 열은 블록의 수이고, 각 행은 블록 안 layer의 수를 뜻합니다~~.~~ 일관된 실험을 위해 고정시킨 매개변수의 값들은 Method 하단에 기재해 놓았습니다. Search space는 convolution layer의 filter 수와 kernel size이며, 실험 결과는 3번 Result에 요약했습니다.

1번의 그래프는, 실험 결과에서 각 알고리즘을 기준으로 정확도와 스파이크 수를 평균 내어 나타낸 것입니다. 그래프에서 보이는 것과 같이, 전반적으로 스파이크가 많을수록 정확도가 높게 나타났습니다. 2번은, 각 조건별로 실험을 진행한 결과를 요약한 표이고, SNN에서 성능이 가장 좋게 나온 것은 가장 아래에 있는 LIF, Soft를 사용한 bayesian 알고리즘 이였습니다. 3번 표는, search space에서 탐색한 값들을 알고리즘 별로 분류해서 정리해 놓은 표입니다.

결과적으로, 세 가지 알고리즘 중에서는 bayesian 알고리즘이, IF보다는 LIF가, Hard보다는 Soft가 더 높은 성능을 보였습니다. 본 연구의 한계점은 자원적인 제한으로 인해 24 GPU 시간밖에 실험을 진행하지 못했기 때문에, greedy 알고리즘 같은 경우 default 값에서 크게 벗어나지 못했다는 점입니다. 향후 연구에서는 더욱 다양한 알고리즘과 긴 시간을 학습해 SNN에 일반적으로 가장 적합한 NAS 알고리즘을 찾고자 합니다.

**[Bayesian]**

매개변수 x에 대한 목적함수 f(x)의 결과값을 최대 또는 최소로 만드는 최적해를 찾는 것을 목적으로 합니다.

목적함수와 hyperparameter 쌍을 대상으로 대체 모델을 만들고, hyperparameter를 순차적으로 업데이트 해가면서 평가를 통해 최적의 hyperparameter 조합을 탐색합니다.

이 때, 목적함수는 black-box function이며 이 함수 f(x)의 결과값을 본 논문에서는 딥러닝 모델의 성능, 즉 검증 정확도로 정했습니다.

대체 모델(**surrogate model**)은, 현재까지 조사된 입력-결과값들에 대한 점들을 바탕으로, 목적 함수에 대한 확률적인 추정을 하는 수행하는 모델입니다.

획득 함수(**Acquisition Function**)는, 이렇게 얻어낸 현재까지의 추정 결과를 바탕으로, 최적 입력값을 찾기에 가장 적합한 다음 입력값 후보를 추천해주는 함수입니다. 본 논문에서는 검증 정확도를 성능의 척도로 삼았기 때문에 bayesian 알고리즘은 검증 정확도를 높이기 위한 매개변수의 조합을 탐색하게 됩니다.

**[Greedy]**

사전 지식을 참고하지 않고 매 선택마다 직전의 평가를 기반으로 바로 앞의 최적의 상황만 쫓아 최종적인 답을 얻는 알고리즘입니다. 때문에 항상 최적의 결과를 얻는 것은 보장되지 않지만, 탐색 속도가 빨라 효율적으로 사용이 가능합니다.