**Revisiting Deep Learning Models for Tabular Data**

**Abstract**

최고의 딥러닝 모델(ResNet-like, FT-transformer)과 gradient Boosted Decision Trees를 비교하고 가장 우월한 솔루션이 없다는 것으로 결론지었다.

**1 Introduction**

정형데이터 모델의 새로운 베이스라인(기준선을 높인)을 제시하기 위해 이미 검증된 아키텍처에서 영감을 얻어 두 가지 모델을 제안한다. ResNet-like 모델과 트랜스포머 구조를 정형 데이터에 맞게 변형한 FT-Transformer 모델을 제안하며, 그후 성능 비교를 위해 다양한 모델들과 비교하고자 한다.

ResNet-like 또한 우수한 모델이고, FT-Transformer도 대부분의 작업에서 최고의 성능을 보여주는 좋은 모델이지만, GBDT와도 비교하였을 때, 아직 보편적으로 우수한 모델은 없다는 결론을 내렸다.

**2 Related Work**

**The “shallow” state of the art**

-정형데이터를 위한 “shallow” state-of-the-art 모델은 ML 대회에서 일반적으로 가장 많이 선택되는 GBDT와 같은 decision tree 앙상블이다

**Differentiable trees**

-결정 트리 앙상블 모델에 영향을 받은 딥러닝 모델인데 GBDT보단 성능이 좋은 경우도 있지만, ResNet-like보단 대부분 안 좋았다.

**Attention-based models**

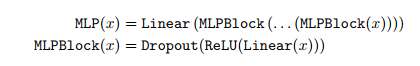
-Attention 기반의 딥러닝 모델들, ResNet-like가 오히려 성능이 좋은 경우가 더 많았다.

**Explicit modeling of multiplicative interactions**

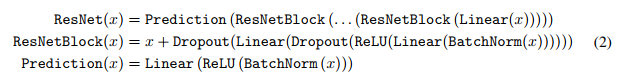
-Multiplicative interaction을 위해 기존 MLP에 feature product를 통합하는 방법을 제안했지만 튜닝된 MLP보다 성능이 좋지 않았다.

**3 Models for tabular data problems.**

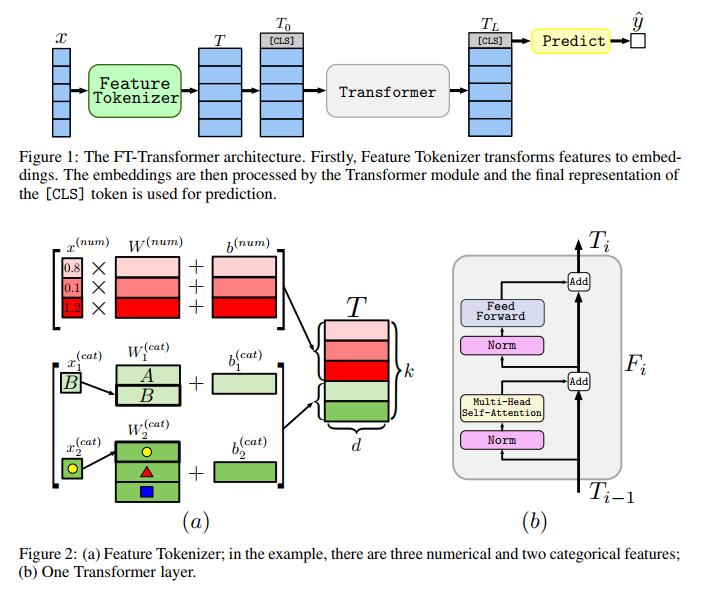
**3.1 MLP**

****

**3.2 ResNet**

****

**3.3 FT-transformer**

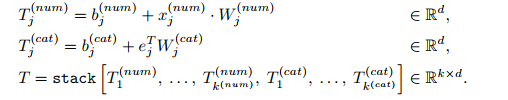
모듈은 크게 두가지로 나뉜다. Feature Tokenizer와 Transformer로 구성되어 있다. ****

**Feature tokenizer**

**FT(Feature tokernizer)는 features인 x를 넣으면 임베딩 값인 T로 변환한다. 아래는 계산식**

****

**(**num)​은 벡터 (num)​에 대한 element-wise multiplication으로 구현하고, (cat)​은 lookup table (cat)​으로 구현한다.

****

**는 범주형 feature과 일치하는 원핫벡터이다.**

**Transformer**

트랜스포머에서는 [CLS] 토큰의 임베딩을 T에 추가함, L개의 Transformer layer를 적용한다.

****

보다 쉬운 최적화를 위해 PreNorm variant를 사용한다. PreNorm 설정에서, 좋은 성능을 위해 첫 번째 Transformer layer에서는 첫 번째 정규화를 제거해야 한다는 것을 발견했다,

**Prediction**

****

**limitation**

FT-Transformer는 ResNet-like 같은 단일 모델보다 훈련에 더 많은 자원을 필요로 하며, 특성의 개수가 너무 많은 데이터셋에는 쉽지 않다. 이러한 문제의 주요 원인은 feature 수에 대한 vanila Multi-Head Self-Attention (MHSA)의 quadratic complexity이 주요 원인이다.

MHSA의 효율에 대한 문제는 MHSA의 근사치를 사용하여 문제를 완화할 수 있다. 추가적으로, 더 나은 추론 성능을 위해 FT-Transformer를 간단한 아키텍처로 바꿀 수 있다.(증류라는 표현을 사용)

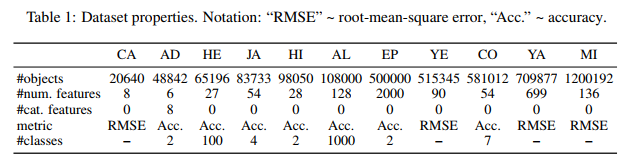
**본 논문에서 사용한 인공지능 모델들: SNN ,NODE,TabNet,GrowNet,DCNV2,AutoInt,XGBoost,CatBoost**

**4.Experiments**

**4.1 scope of the comparison**

**이 논문에서는 모델 성능의 상대적인 비교에 집중하며, 사전학습이나 추가적인 손실함수등과 같은 model-agnostic DL-practices는 신경쓰지 않는다.**

**4.2 Datasets**

****

**4.3 implementation details**

**Data Preprocessing-**공정한 비교를 위해 모두 동일한 전처리 방법을 사용한다. 본적으로 사이킷런의 quantile transformation을 사용하였고, Helena와 ALOI에는 표준화(mean subtraction and scaling)를 적용한다. Epsilon의 경우, 데이터셋을 그대로 사용하였다. 모든 알고리즘의 회귀 타켓은 표준화를 적용한다.

**Tuning-** 모든 데이터셋에 대해 각 모델들의 하이퍼파라미터를 조정한다,

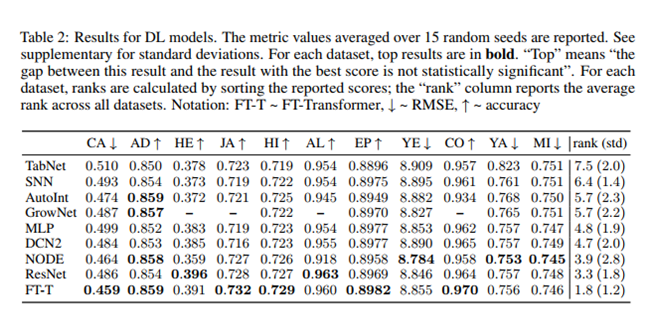
**Evaluation-** 각 튜닝 구성에서 우리는 각각 다른 랜덤 시드로 15번의 실험을 진행하고 테스트 셋에서의 결과를 기록한다. 몇몇 알고리즘은 우리는 튜닝을 하지 않은 기본구성에서의 결과도 기록한다,

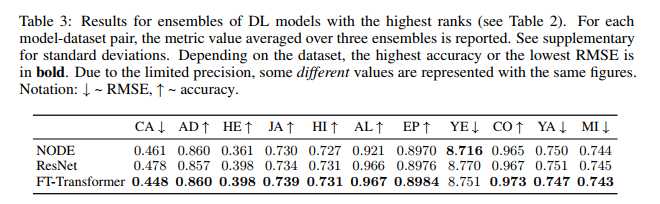
**Ensembles-**각 모델마다, 각 데이터셋에서 우리는 세 emsemble를 얻음 15개 단일 모델을 3그룹으로 찢어서 긱 그룹에서 모델들의 평균을 구한다.

**Neural Networks-** 분류 문제에는 크로스 엔트로피를, 회귀 문제에는 mse를 사용한다. TabNet과 GrowNet의 경우 원래의 구현과 Adam optimizer를 사용한다. 나머지 알고리즘의 경우 AdamW optimizer를 사용한다. 또한, learning rate schedules를 사용하지 않는다.

각 모델의 논문에서 배치 사이즈에 대한 특별한 지침이 있지 않은 한, 모든 알고리즘에 미리 정의된 배치 크기를 사용함. 검증 데이터셋에 대한 개선이 patience + 1 이상 없다면 학습을 종료하며, 모든 알고리즘에 대해 patience = 16로 설정함.

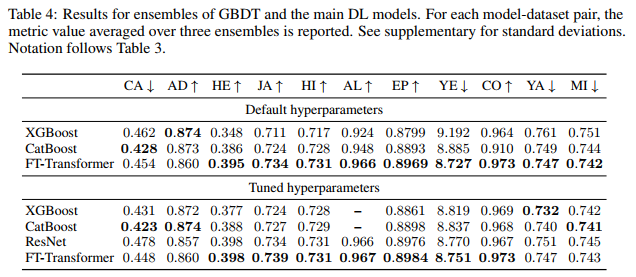
**Categorical features-** XGBoost에서는 원 핫 인코딩을 사용 Catboost에서는 우리는 내부적인 지원(보유 기능)을 사용, 신경망에서는 모든 범주형 Features에 대해 같은 차원의 embedding을 진행

**4.4 Comparing DL models**

****

**4.4 Comparig DL models and GBDT**

GBDT는 Ensemble 기술이 쓰이므로 단일모델들 대신에 Ensemble 모델로 비교를 했다.

****

**기본 하이퍼파라미터:**

FT-TRANSFORMER 앙상블이 대부분 GBDT에 비해 좋은 성능을 보여준다. 흥미롭게도 FT-TRANSFORMER 기본 ensemble은 튜닝된 ensemble과 상당히 유사하게 수행한다**.**

**FT-transformer는 기본상태의 강력한 앙상블이 사용가능하다**

**튜닝 된 하이퍼파라미터:**

일단 하이퍼파리미터들이 적절하게 튜닝되면, GBDT들은 몇몇 데이터셋에서 뛰어난 성능을 보이게 되고, DL모델들이 더 성능이 좋은 경우에도 데이터셋 문제라고도 볼 수 있다. GBDT는 클래스가 많은 다중 분류 문제에는 여전히 부적절하다. 클래스 수에 따라 GBDT는 만족스럽지 못한 성능을 보이거나(Helena), 매우 느린 훈련으로 하이퍼파라미터의 튜닝이 불가능할 수도 있다(ALOI).

**딥러닝과 GBDT에서 보편적인 해결책은 없다.**

**DL friendly 문제 해결이 필요하다**

**4.6 An intriguing property of FT-Transformer**

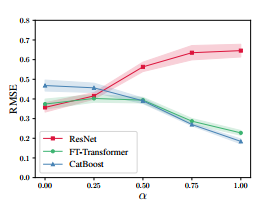
FT-Transformer는 **보편적으로 좋은 성능을 제공**하는 반면, GBDT와 ResNet-like 모델은 **일부 하위 모델들에서만 좋은 성능**을 보여주었다.

**5, Analysis**

**5.1 When FT-Transformer is better than ResNet?**

**실험을 통해 FT-Transformer가 어떤 데이터셋에서도 성능이 꾸준히 좋은 모델이라는걸 보여주었다.**

FT-Transformer와 ResNet-like 모델 간 성능 차이를 이해하기 위해 실험을 진행했다. ****

****

는 GBDT에 유리하고 은 ResNet에 유리하다

는 무작위로 구성된 30개의 결정 트리의 평균 예측값이며, 은 무작위로 초기화된 3개의 hidden layer를 가진 MLP의 예측값이다.

ResNet-friendly 타겟에서, ResNet-like 모델과 FT-Transformer는 비슷한 성능을 보였으며 CatBoost 보다 성능이 우수하였음.

타겟이 GBDT-friendly가 되면 ResNet-like 모델의 상대적 성능이 크게 떨어짐.

FT-Transformer는 전체 작업 범위에서 경쟁력 있는 성능을 제공하였음.

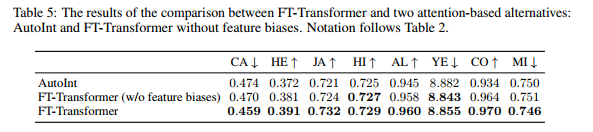
**5.2 Ablation study**

**이 실험을 통해 FT-transformer가 다른 attention모델들에 비해 뛰어남을 보여주었다**.

첫번째 우리는 FT-transformer와 비슷한 모델인 Autoint모델과 비교를 했다. AutoInt 또한 모든 feature들을 임베딩하고 self-attention을 적용했다. 하지만 AutoInt의 embedding layer는 feature biases를 포함하지 않는다. 그것의 backbone은 vanila transformer와 다르다. 그리고 추론 메커니즘도 CLS TOKEN을 사용하지 않는다.

두번째는 좋은 수행능력을 위해서 FT속 feature bias가 필수인지 아닌지를 확인했다.

실험결과 Transformer의 backbone이 AutoInt에 비해 우월하고, feature biases가 필요하다라는 결과를 도출하였다.

****

**5,3 Obtaining feature importance from attention maps**

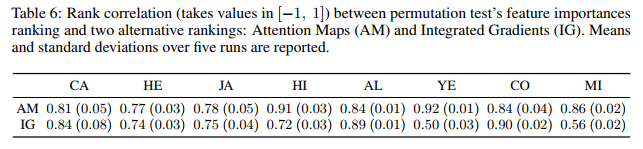
**Attention map 기반 방법은 합리적인 특성 중요도를 산출하였고, Integrated Gradients와 비슷하게 작동하였다. IG의 느린 계산 속도와 계산 비용을 고려할 때, 어탠션 맵이 좋은 선택이다.**

Attention maps으로부터 feature의 중요성을 얻을 수 있을지 실험함 

i번째 샘플에서 평균 어텐션맵인 를 계산하고 이를 통해 특성 중요도를 나타내는 분포인 p를 구한다.

​은 *i*번째 샘플의 *l*번째Transformer layers, *h*번째 헤드의 [CLS] 토큰의 attention map이다.

이러한 접근 방법을 평가하기 위해, attention map과 Integrated Gradients(IG)를 비교한다. 모든 데이터셋에 permutation test를 수행하여 모델의 해석 능력을 확인한 결과는 아래와 같다.

****

**6 Conclusion**

**FT-transformer가 성능이 우수한 모델이라는 걸 보여주었지만, GBDT와 비교했을 때 압도적인 성능을 보여주진 못했다.**

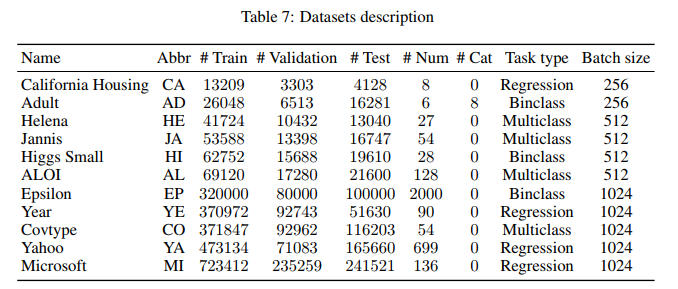
**Supplementary material**

**논문 실험 구현을 위해 필요한 내용이 담겨있습니다**

**A Software and hardware**

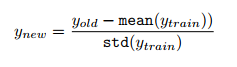
모든 실험들은 같은 조건의 소프트웨어 상에서 수행된다. 실험에서 사용된 하드웨어는 소스코드에서 찾아볼 수 있다.

**B Data**

****

**b.2 preprocessing**

회귀문제를 위해 우리는 타켓값(종속변수)을 표준화한다.

****

quantile preprocessing의 파라미터들을 계산하기 위해 평균이 0이고 분산이 1e-3인 정규분포를 따르는 노이즈들을 학습시킬 연속형 변수에 추가한다. GBDT는 전처리하지 않는다.

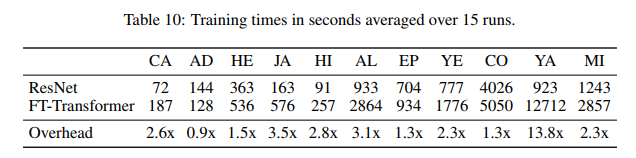
**C results for all algorithms on all datasets**

메인 텍스트에서 통계적 중요도를 측정하기 위해 p값 0.01하에서 one-sided wilcoxin 테스트를 진행했다.

**D Additional results**

**D.1 Training times**

학습시간이 길다고 모델 성능이 좋아지지는 않는다는 것을 증명했고, 실험내용은 아래와 같다.

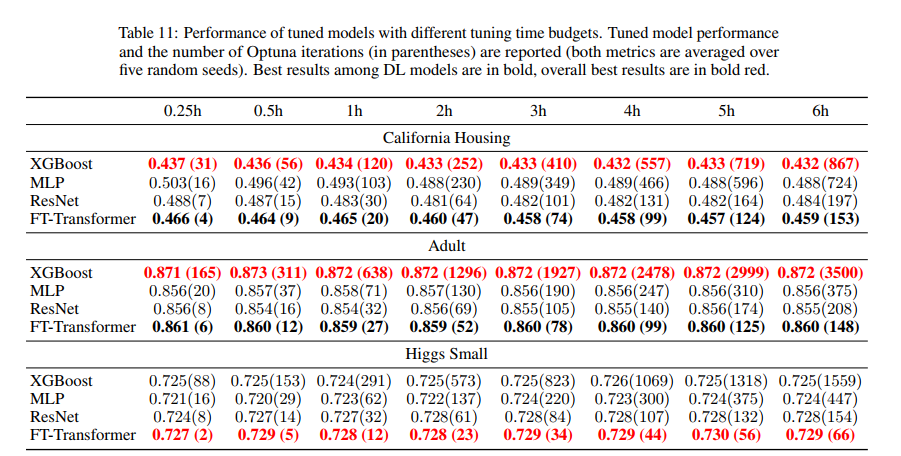
****

대부분의 실험에서 학습시간은 소스코드에서 찾을 수 있다. 오버헤드 시각화 위해 표 작성

야후 데이터에서 시간 차이가 가장 큰 이유는 feature 개수(700개)가 가장 크기 때문이다

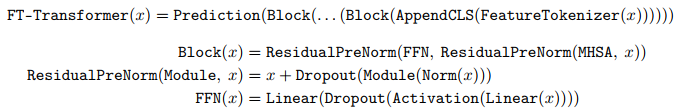
* 학습시간을 튜닝하는것에 따른 튜닝된 모델들의 상대적 성능차이가 어떻게 되는가?
* iteration 튜닝개수가 모델의 잠재력을 올려주는가?

첫번째 질문은 두가지 이유로 중요히다. 첫번째 우리는 긴 학습시간이 강한 성능의 원인이 되지않는다는걸 확인해야 한다. 두번째로 우리는 ft-transformer를 적은 시간 내에 테스트 하기를 원한다.

아래 표와 같이 각 데이터셋마다 5번의 독립적인 하이퍼파라미터 최적화를 수행한다.15분, 30분, 1시간, 2시간, 3시간, 4시간, 5시간, 6시간 동안 하이퍼파라미터를 튜닝한 뒤 성능을 확인하고, 모델의 성능과 함께 5개 random seed에 대한 평균값으로 Optuna iteration 또한 확인한다**.**

**E FT-Transformer**

**E.1 Architecture**



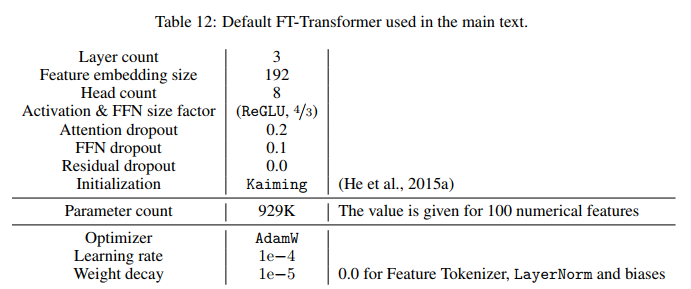
정규화를 위해 LayerNorm를 사용하였으며, MHSA에서는 nheads​=8을 적용함. ReGLU 활성화함수를 사용하였고, 간단하고 동일한 최적화를 위해 PreNorm을 사용했다.(Transformer 원본은 PostNorm을 사용했다.)

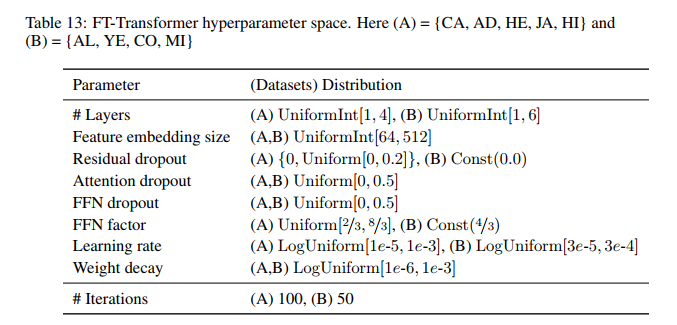
활성화-RegLU활성화 함수를 사용

dropout비율-attention dropout은 유익하고, FFN dropout은 0이 아닌 값으로 보통 세팅되며 최종 dropout은 0이 아닌 값으로 세팅되는 것이 드물다.

**PreNorm vs PostNorm- 간단하고 동일한 최적화를 위해 PreNorm을 사용, original transformer은 postnorm을 사용**

**E.2 The default configuration**

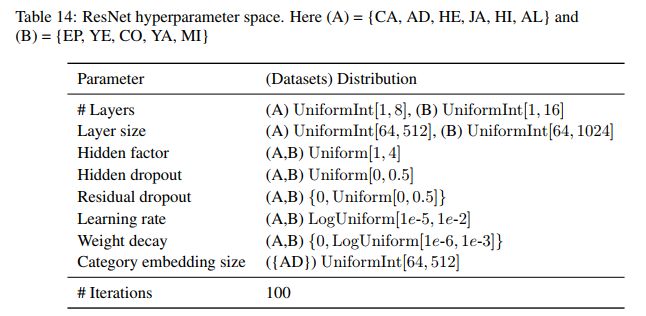
****

****

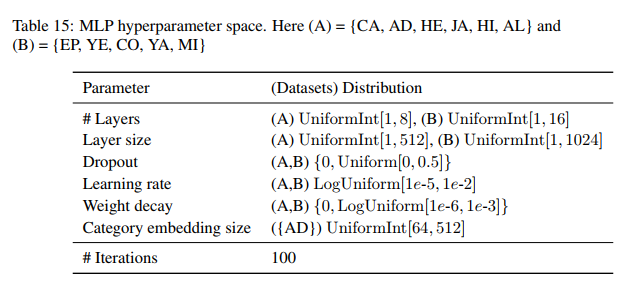
**F models**

모델 디테일(파라미터 정리)

**F.1 ResNet**

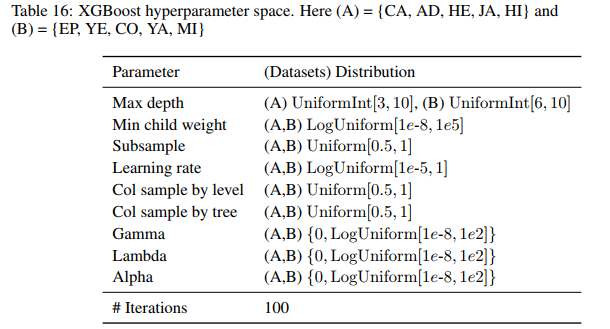
****

**F.2 MLP**

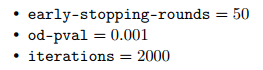
****

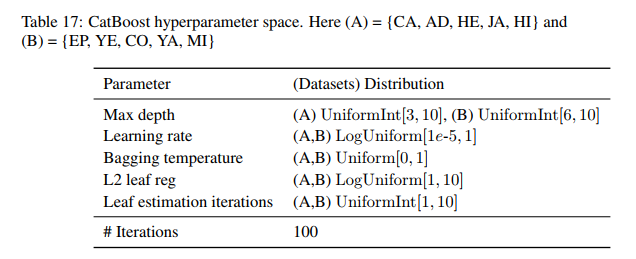
**F,3 XGBOOST**

****

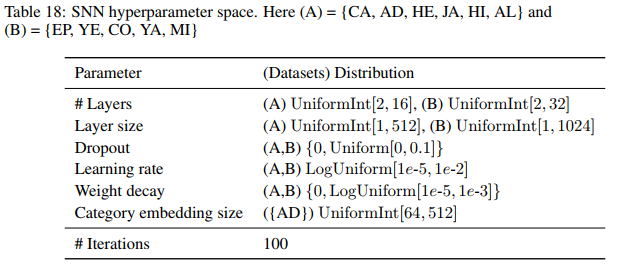
****

**F.4 CatBoost**

****

****

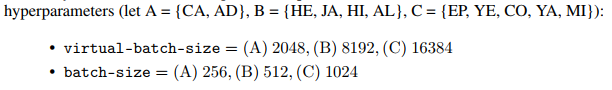
**F.5 SNN**

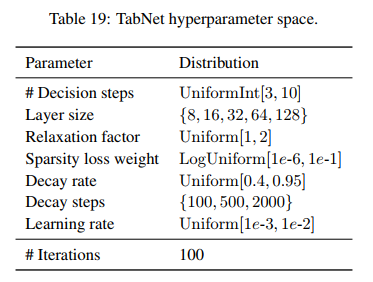
****

**F.6 NODE**

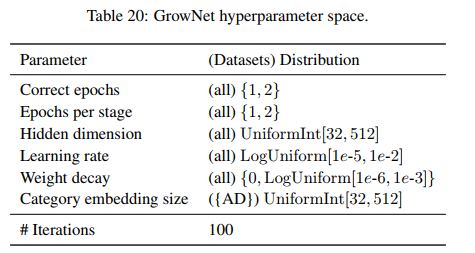
**튜닝시 노드는 class수가 많으면 분류문제로 측정할 수 없음을 주의해야 한다.**

**F.7 TabNet**

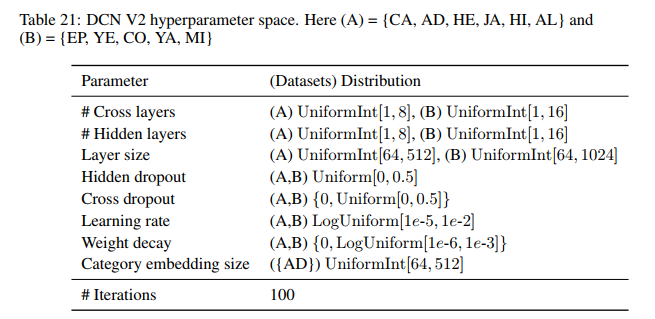
****

****

**F.8 GrowNet**

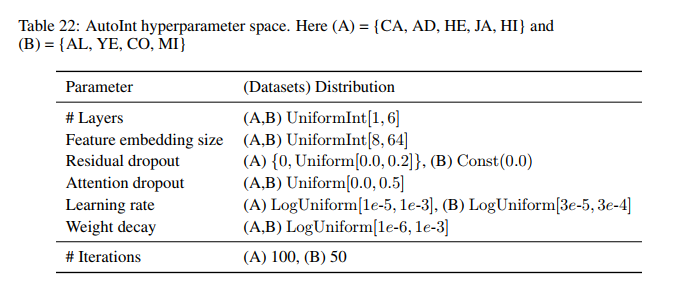
****

**F.9 DCN V2**

****

**F.10 AutoInt**

**nheads=2**

****

**G. analysis**

**G.1 When FT-Transformer is better than ResNet?**

**Data**-훈련데이터 500000, 검증데이터 50000 테스트데이터 100000

- 세개의 은닉층으로 이루어진 mlp로 구현되었다. 각 사이즈는 256개고 가중치는 kaiming initialization으로 설정되어있다. bias들은 아래와 같은 유니폼분포를 따른다. 모든 파라미터들은 초기화후 고정된다.



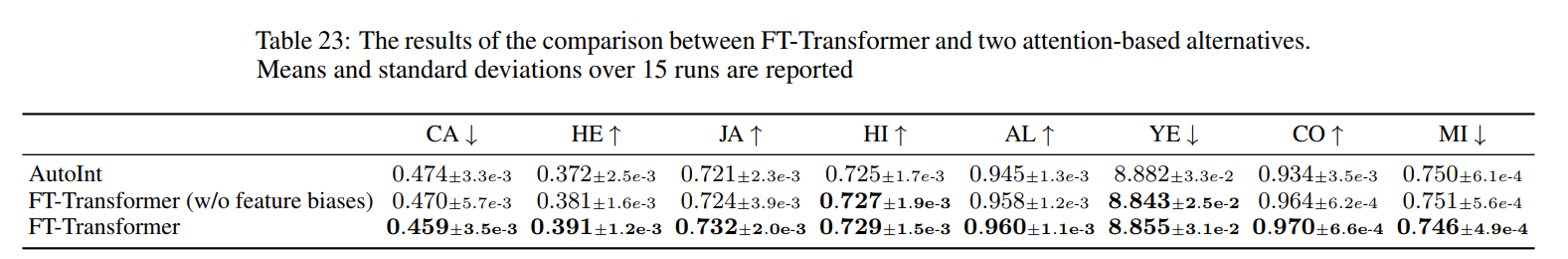
- 이 함수는 30개의 랜덤한 결정트리의 평균 예측값으로 구현되었다. 한 결정 트리의 구조는 알고리즘 1을 보여준다. 하나의 결정 트리의 추론과정은 평범한 결정 트리들과 같다.

**Catboost**-기본 하이퍼파라미터 사용

**FT-transformer**-기본 하이퍼파라미터 사용, parameter count: 930k

**ResNet**-Residual block count:4 , embedding size: 256, Dropout rate inside residual blocks: 0.5 parameter count: 820k

**G.2 Ablation study**

****