Neural Lineage

2024년도 하계 세미나

Sogang University
Vision & Display Systems Lab, Dept. of Electronic Engineering

Presented By
ChanHee Kang

Outline

- Background
  - Neural Tangent Kernel: Convergence and Generalization in Neural Networks (NeurIPS 2018)
- Paper
  - Neural Lineage (CVPR 2024 oral)

Background

- **Neural Tangent Kernel(NTK)**

  - **신경망의 학습 과정을 이해하기 위한 도구**

    - **NTK는 신경망(ANN)의 width가 무한대로 커질 때, 신경망의 동작을 기술 할 수 있는 커널 함수**

      - **커널 함수**

        ✓**두 입력 사이의 유사도를 측정하는 함수**

        ✓**커널 함수는 입력 데이터를 고차원 특징 공간으로 매핑하여, 그 공간에서 선형적으로 분리 가능한 패턴을 찾을 수 있게 함**

        ✓**실제로 고차원 공간으로의 매핑을 명시적으로 계산하지 않아도, 커널 함수를 통해 동일한 효과를 얻을 수 있음. 이를 'kernel trick'이라고 함**

      - **커널 함수의 예시**

        ✓**RBF(Radial Basis Function) 커널은 두 점 사이의 거리 기반 유사도를 계산**

        - $k(x, x^*) = e^{-\frac{||x-x^*||^2}{2\sigma^2}}$

Background

- **Neural Tangent Kernel(NTK)**

  - **신경망의 학습 과정을 이해하기 위한 도구**

    - **NTK는 신경망(ANN)의 파라미터가 무한대로 커질 때, 신경망의 동작을 기술 할 수 있는 커널 함수**

      - 이를 통한 분석에 따르면, ANN의 width가 증가함에 따라, 네트워크가 1차 테일러 다항식에 가까워진다는 결론을 얻을 수 있음

      - NTK 뿐만 아니라 다양한 분야에서의 연구가 네트워크의 1차 근사식이, 매우 작은 sample을 가진 영역이나, 특별하게 디자인된 문제에서 비선형 네트워크보다 더 좋은 성능을 보인다는 사실을 발견함

      - 이러한 1차 근사를 통하여 ANN을 근사하는 방식을 “Neural Network Linearization”이라고 함

        ✓이러한 linearization은 training free neural architecture search 나 training time prediction, deep learning model ensemble에서 사용되는 기법 중 하나임

Neural Lineage Detection

- **Neural Lineage(CVPR 2024)**
  - **Neural Lineage detection**
    - 딥러닝 모델 간의 parent-child 관계를 탐지하는 새로운 task인 neural lineage detection을 제안
      - Neural lineage detection의 목적은 주어진 child model이 어떤 parent model에서 finetuning 되었는지를 예측하는 것
    - 이러한 task를 해결하기 위한 두 가지 방법을 제안
      - Learning free approach
        - ✓Approximation step: NN linearization을 사용하여 finetuning 과정 근사
        - ✓Measurement step: 근사된 자식 모델과 실제 자식 모델 간의 거리를 다양한 유사성 metric을 사용하여 비교
        - ✓이 방법은 NTK 이론을 기반으로 하며, 여러 유사성 metric과 finetuning 과정을 근사하는 과정을 통합

Neural Lineage Detection

- **Neural Lineage(CVPR 2024)**
  - **Neural Lineage detection**
    - 딥러닝 모델 간의 parent-child 관계를 탐지하는 새로운 task인 neural lineage detection을 제안
      - Neural lineage detection의 목적은 주어진 child model이 어떤 parent model에서 finetuning 되었는지를 예측하는 것
    - 이러한 task를 해결하기 위한 두 가지 방법을 제안
      - Learning based approach
        - ✓Weight encoder, feature encoder 및 transformer detector로 구성됨
        - ✓부모 및 자식 모델의 weight, feature를 인코딩 하고, transformer를 이용하여 확률을 예측

Neural Lineage Detection

- **Neural Lineage(CVPR 2024)**
  - **Neural Lineage detection이 필요한 이유**
    - **딥러닝 모델의 진화 및 변화**
      - **모델 간의 의존성 증가**
        - ✓최근 딥러닝 모델들은 더 이상 독립적으로 존재하지 않고, 사전 학습된 모델을 기반으로 한 finetuning이 일반화 되었음
        - ✓따라서 다양한 응용 분야에서 이러한 의존 관계를 이해하는 것이 중요
      - **모델 네트워크의 확장**
        - ✓딥러닝 모델의 수가 급증하고 있으며, 다양한 모델 간의 세대 간 관계가 형성되고 있음
        - ✓이는 모델들이 서로 어떻게 연결되고 진화하는지 이해할 필요성이 있음을 의미함

Neural Lineage Detection

- **Neural Lineage(CVPR 2024)**

  - **Neural Lineage detection이 필요한 이유**

    - **실제 적용 사례의 중요성**

      - **모델 재사용**

        ✓**Lineage detection을 이용하여 모델의 knowledge hierarchy를 이해하고, 모델의 generalization 및 robustness를 평가할 수 있음**

        - 부모 모델에서 자식 모델로 상속된 지식의 양과 질을 평가할 수 있음
        - 예를 들어, 부모 모델이 다양한 데이터셋에서 높은 성능을 보였을 경우, 자식 모델도 비슷한 일반화 성능을 보일 가능성이 높음
        - 반대로, 특정한 편향을 가진 부모 모델에서 finetuning된 자식 모델도 유사한 편향을 가질 수 있으므로, 이를 조기에 발견하고 수정할 수 있음
        - 모델의 계보(lineage)를 추적함으로써, 다양한 환경과 데이터셋에서 모델이 어떻게 작동하는지 평가할 수 있음. 이는 모델의 robustness를 평가하는데 중요한 역할을 함
        - 예를 들어, 다양한 부모 모델에서 finetuning된 자식 모델들이 특정 상황에서 모두 높은 성능을 보인다면, 이는 모델이 해당 상황에서 robust 하다는 것을 의미

Neural Lineage Detection

- **Neural Lineage(CVPR 2024)**
  - **Neural Lineage detection이 필요한 이유**
    - **실제 적용 사례의 중요성**
      - **모델 재사용**

✓**Lineage detection을 이용하여 모델의 knowledge hierarchy를 이해하고, 모델의 generalization 및 robustness를 평가할 수 있음**

- **부모 모델과 자식 모델의 관계를 이용하며, 자식 모델의 성능 저하 원인을 분석할 수 있음**
- **예를 들어, 부모 모델에서 finetuning된 자식 모델들이 공통적으로 성능 저하를 보일 경우, 그 원인을 부모 모델에서 찾을 수 있음**

Neural Lineage Detection

- **Neural Lineage(CVPR 2024)**

  - **Learning free approach**

    $\bar{f}_p$: 선형 근사된 parent 모델
    $\theta_c - \theta_p$: child 모델과 parent 모델 간의 파라미터 변화
    N: data sample의 개수
    K: target dimension

    - **Linearized parent model (이론적 근거)**

$$\bar{f}_p(x) \coloneqq f_p(x) + \nabla_{\theta_p} f_p(x)(\theta_c - \theta_p)$$

파라미터가 θ_p 에서 θ_c로 변할 때, 그 변화가 입력 x 에 대한 출력에 미치는 영향을 구하게 됨

  - **Similarity metric**

    - **l_p similarity**

$\Pi_i = -\frac{p}{NK}\left[sign(d_i) * |d_i|^{p-1}\right]^T$ for L_p similarity
$d_i \coloneqq f_p(x^{(i)}) - f_c(x^{(i)})$

$$s(\bar{f}_p, f_c) \coloneqq -\frac{1}{NK}\sum_{i=1}^{N}\left\|\bar{f}_p(x^{(i)}) - f_c(x^{(i)})\right\|_p^p$$

    - **Linear approximation form of similarity**

$$s(\bar{f}_p, f_c) \approx s(f_p, f_c) + \nabla_{\theta_p}\left[\sum_{i=1}^{N} sg(\Pi_i) f_p(x^{(i)})\right](\theta_c - \theta_p)$$

Neural Lineage Detection

- **Neural Lineage(CVPR 2024)**

  - **Learning free approach**

    $\bar{f}_p$: 선형 근사된 parent 모델
    $\theta_c - \theta_p$: child 모델과 parent 모델 간의 파라미터 변화
    N: data sample의 개수
    K: target dimension

    - **Linearized parent model (이론적 근거)**

$$\bar{f}_p(x) \coloneqq f_p(x) + \nabla_{\theta_p} f_p(x)(\theta_c - \theta_p)$$

파라미터가 θ_p 에서 θ_c로 변할 때, 그 변화가 입력 x 에 대한 출력에 미치는 영향을 구하게 됨

  - **Similarity metric**

    $\Pi_i = -\frac{1}{NKt}[sign(d_i) * softmax(t|d_i|)]^T$ for LSE similarity
    $d_i \coloneqq f_p(x^{(i)}) - f_c(x^{(i)})$

    - **Log-sum-exp similarity**

$$s(\bar{f}_p, f_c) \coloneqq -\frac{1}{NKt}\sum_{i=1}^{N} log \sum_{k=1}^{K} e^{t|\bar{f}_p(x^{(i)}) - f_c(x^{(i)})|}$$

    - **Linear approximation form of similarity**

$$s(\bar{f}_p, f_c) \approx s(f_p, f_c) + \nabla_{\theta_p}\left[\sum_{i=1}^{N} sg(\Pi_i) f_p(x^{(i)})\right](\theta_c - \theta_p)$$

Neural Lineage Detection

- **Neural Lineage(CVPR 2024)**

  - **Learning free approach**

    $\bar{f}_p$: 선형 근사된 parent 모델
    $\theta_c - \theta_p$: child 모델과 parent 모델 간의 파라미터 변화
    N: data sample의 개수
    K: target dimension

    - **Linearized parent model (이론적 근거)**

$$\bar{f}_p(x) \coloneqq f_p(x) + \nabla_{\theta_p} f_p(x)(\theta_c - \theta_p)$$

파라미터가 θ_p 에서 θ_c로 변할 때, 그 변화가 입력 x 에 대한 출력에 미치는 영향을 구하게 됨

  - **Similarity metric**

$$K = XX^T$$
$$L = YY^T$$
$$K_c = K - 1K - K1 + 1K1$$

X: 표현집합(여기서는 parent model output)
Y: 표현집합(여기서는 child model output)
1: 모든 element가 1인 행렬을 element의 수로 나눈 행렬

    - **Centered Kernel Alignment(CKA)**

      - NN 표현의 유사성을 측정하는 지표 중 하나
      - Parent, child 모델의 output을 이용하여 kernel matrix를 구하고, 구한 행렬의 각 행과 열의 평균을 빼고 전체 평균을 더하여 중심화된 커널 행렬 구함
      - 중심화된 커널 행렬 간의 유사성을 측정하는 Hilbert-Schmidt Independence Criterion(HSIC) 계산

$$CKA(K, L) = \frac{HSIC(K_c, L_c)}{\sqrt{HSIC(K_c, K_c) \cdot HSIC(L_c, L_c)}}$$

Neural Lineage Detection

- **Neural Lineage(CVPR 2024)**

  - **Learning free approach**

    - **Linearized parent model (이론적 근거)**

$\bar{f}_p$: 선형 근사된 parent 모델
$\theta_c - \theta_p$: child 모델과 parent 모델 간의 파라미터 변화
N: data sample의 개수
K: target dimension

$$\bar{f}_p(x) \coloneqq f_p(x) + \nabla_{\theta_p} f_p(x)(\theta_c - \theta_p)$$

파라미터가 θ_p 에서 θ_c로 변할 때, 그 변화가 입력 x 에 대한 출력에 미치는 영향을 구하게 됨

  - **Similarity metric**

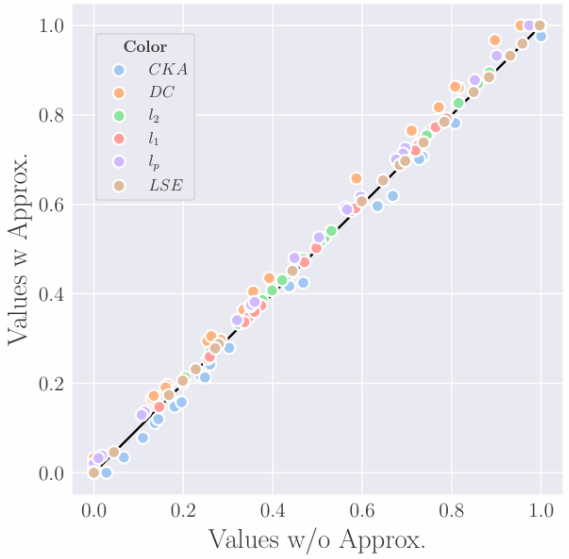
    - **Squared Distance Correlation(DC)**

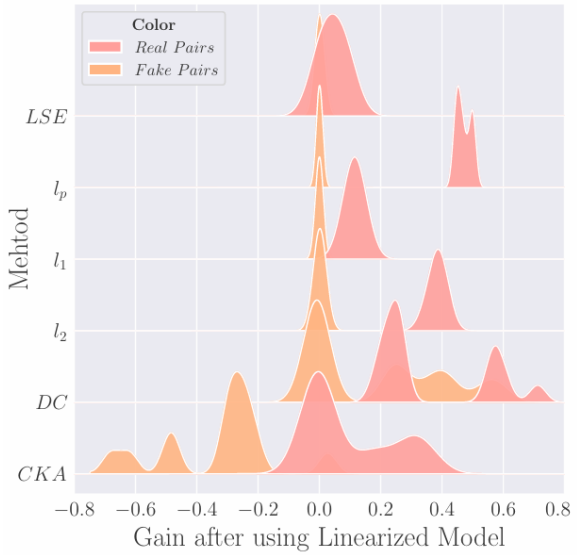
      - X, Y 각각에 대하여 거리 행렬 A, B를 계산
      - 거리 행렬 A의 element a_{ij} 는 데이터 X_i, Y_j간의 거리를 의미
      - 중심화된 거리 행렬 A_c, B_c계산

$$dCor^2(X,Y) = \frac{HSIC(A_c, B_c)}{\sqrt{HSIC(A_c, A_c) \cdot HSIC(B_c, B_c)}}$$

Neural Lineage Detection

- **Neural Lineage(CVPR 2024)**
  - **Learning free approach**
    - **Approximation 방법의 정당성을 위한 실험**
      - **Linearization 전 후의 metric의 차이가 거의 없음 (y=x 에 분포) (a)**
      - **Linearization 전 후의 gain의 분포가 더 좋음(Real pair 끼리의 metric이 양의 값 가짐) (b)**





Neural Lineage Detection

- **Neural Lineage(CVPR 2024)**
  - **Learning free approach**
    - **Approximation 방법의 정당성을 위한 직관적인 설명**

$$f'_p(x) \coloneqq f_p(x) + W(x)\left(\theta_c - \theta_p\right)$$

$$f''_p(x) \coloneqq f_p(x) + \nabla_{\theta_p} f_p(x) Z$$

For real parent-child model pair (f_p, f_c) and l_2 similarity,

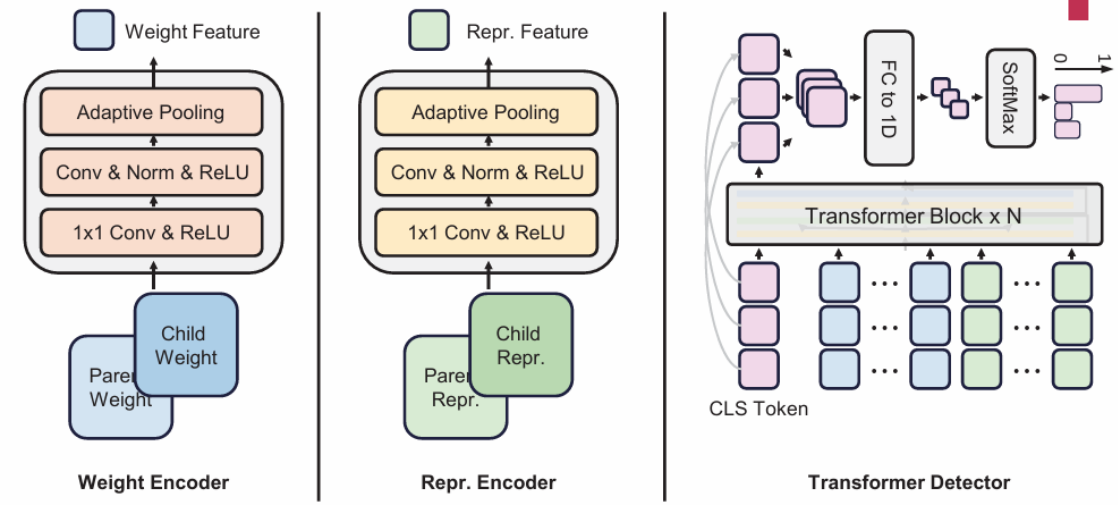
$$s\left(\bar{f}_p, f_c\right) = \max_{W(x), x \in \chi} s(f'_p, f_c) = \max_{Z} s(f''_p, f_c)$$

이므로 W, Z에 대한 적합한 값이, linearization metho에서 선택한 값과 같음을 알 수 있다.

Neural Lineage Detection

- **Neural Lineage(CVPR 2024)**
  - **Learning based approach**
    - **Weight and feature encoder**
      - 1x1 convolution 연산을 통해 element-level relationship
      - AP는 output size가 1x1이 되도록 강제

```python
def forward(self, weight, feature):
    # Nxnumpx2xwxh
    weightshape = weight.shape
    assert self.nump == weightshape[1]
    weight = weight.view(weightshape[0]*weightshape[1], *weightshape[2:]) # (Nxnump)x2xwxh
    featureshape = feature.shape
    assert self.nump == featureshape[1]
    feature = feature.view(featureshape[0]*featureshape[1], *featureshape[2:]) # (Nxnump)x2xwxh

    weight = self.weightEnc(weight) # (Nxnump)xE
    feature = self.featureEnc(feature) # (Nxnump)xE

    weight = weight.squeeze(-1).squeeze(-1)
    feature = feature.squeeze(-1).squeeze(-1)

    weight = weight.unsqueeze(1) # (Nxnump)x1xE
    feature = feature.unsqueeze(1) # (Nxnump)x1xE

    concatenated = self.tokenization(weight, feature) # (Nxnump)x3xE
    attn_output,_ = self.attention(concatenated,concatenated,concatenated,need_weights=False) # (Nxnump)x3xE
    attn_output = attn_output[:, 0, :] # (Nxnump)xE

    attn_output = attn_output.view(weightshape[0], self.nump, attn_output.shape[-1])

    attn_mean = attn_output.mean(dim = (1,2),keepdim = True)
    attn_std = attn_output.std(dim = (1,2),keepdim = True)
    attn_output = (attn_output - attn_mean)/attn_std

    if self.noparent:
        noparent_token = self.noparent_token.expand(weightshape[0],-1,-1)
        attn_output = torch.cat((attn_output, noparent_token), dim = 1)

    output = self.fc(attn_output)
    return output.squeeze(-1)
```



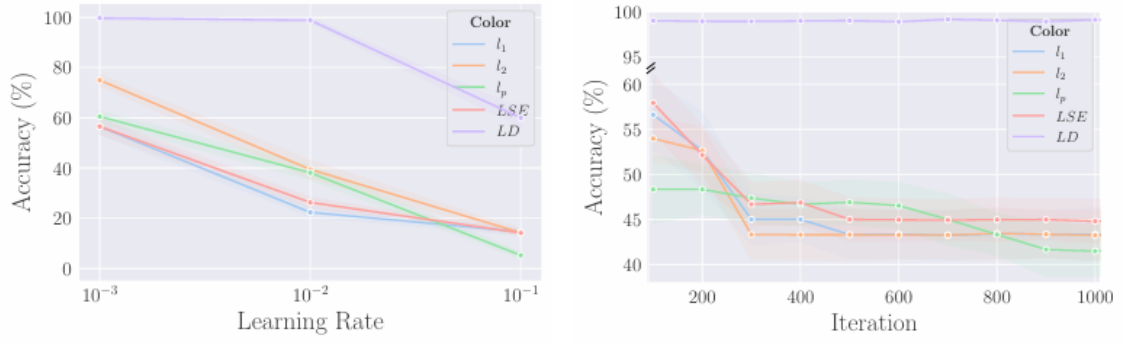
Neural Lineage Detection

- **Neural Lineage(CVPR 2024)**
  - **Learning based approach**
    - **실험 결과(Classification) (Metric: average accuracy)**

			FMNIST	EMNIST	Cifar10	SVHN	DTD	Pet	Cifar10 5shot	Cifar10 50shot	Cifar10 Imbalanced
Fully Connected Network	w/o Approx.	l_1	24.11±1.27	35.97±1.25	70.04±1.05	84.09±1.64	-	-	80.31±1.22	73.93±1.01	75.42±1.43
		l_2	23.98±2.04	45.61±1.59	76.88±0.64	68.84±1.71	-	-	86.37±1.02	75.32±1.18	78.24±1.72
		l_∞	11.84±1.23	37.99±1.83	77.32±0.81	70.06±2.25	-	-	78.79±1.44	75.14±1.08	76.41±1.24
		CKA	4.39±0.37	6.58±0.76	52.49±1.31	29.92±1.27	-	-	18.18±1.83	21.21±0.97	50.76±1.72
		DC	3.51±0.49	6.32±0.91	54.92±1.22	29.92±0.87	-	-	19.71±1.71	21.81±1.52	50.38±1.61
	w Approx.	l_1	25.44±1.27	36.40±1.31	71.59±1.12	85.22±1.57	-	-	85.61±2.51	74.62±1.12	75.76±1.51
		l_2	26.75±1.75	46.93±1.73	80.03±0.71	70.07±1.71	-	-	88.64±1.31	75.38±1.18	78.79±1.72
		l_p	21.05±1.69	41.23±1.79	78.79±0.83	70.83±1.61	-	-	86.73±1.41	69.71±1.51	76.51±1.16
		LSE	26.32±1.27	37.28±1.36	73.48±0.78	71.59±1.79	-	-	85.61±1.39	75.76±1.05	78.03±1.47
		CKA	5.71±0.48	14.91±1.04	53.03±1.16	57.95±0.98	-	-	21.21±2.41	21.97±0.96	51.14±1.69
		DC	14.48±1.41	7.64±0.81	61.74±0.91	28.41±1.01	-	-	25.01±2.42	36.74±1.89	51.14±1.41
	Lineage Detector		**97.86±1.19***	**93.61±2.37***	**99.11±0.36***	**99.31±0.35***	-	-	**99.94±0.35***	**98.84±1.43***	**99.35±0.24***
ResNet18	w/o Approx.	l_1	90.37±0.82	90.47±0.84	90.19±1.33	98.75±0.72	89.41±1.66	90.23±0.64	98.85±0.19	98.68±0.88	87.50±1.11
		l_2	86.84±1.15	87.51±1.36	90.10±1.41	98.64±0.22	88.68±1.66	89.43±0.99	97.75±0.19	98.41±0.41	87.68±0.11
		l_∞	78.57±0.87	95.83±0.65	79.76±0.64	95.24±0.41	88.24±2.01	85.81±1.45	98.07±0.69	98.93±0.51	79.17±1.21
		CKA	68.45±2.49	75.59±2.19	45.83±0.81	63.54±1.93	62.94±1.52	66.04±1.98	71.11±3.45	64.82±0.52	62.51±1.48
		DC	58.92±2.21	32.14±1.61	61.33±2.71	45.43±1.68	62.94±1.75	51.47±3.17	67.78±2.28	65.56±0.51	60.47±1.59
	w Approx.	l_1	98.81±0.41	98.92±0.59	94.64±0.63	98.81±0.41	94.71±0.32	95.27±0.84	99.89±0.62	98.93±0.61	95.84±0.99
		l_2	88.69±0.65	88.09±1.08	90.47±0.84	97.94±1.26	90.59±1.75	90.53±0.63	98.89±0.59	99.36±0.52	88.09±0.93
		l_p	80.95±0.48	80.95±1.16	81.55±0.68	94.05±0.92	91.18±1.87	89.66±0.35	92.22±1.86	93.54±2.23	80.95±0.85
		LSE	96.43±0.97	98.21±0.41	89.29±1.21	99.41±0.33	13.53±1.43	11.24±1.21	98.82±0.22	97.85±1.21	91.67±1.23
		CKA	70.23±2.54	77.38±1.93	47.62±0.94	64.88±1.89	66.47±0.83	68.05±1.61	72.22±3.25	68.81±3.71	66.08±2.42
		DC	60.11±2.33	34.52±0.85	63.09±2.73	45.83±1.47	63.12±1.43	52.07±2.73	67.89±2.18	65.58±2.64	60.71±1.09
	Lineage Detector		**99.21±0.16**	**99.32±0.05**	**99.87±0.07***	**99.64±0.12**	**99.59±0.47***	**99.74±0.21***	**99.91±0.52**	**99.38±0.02**	**97.01±0.41**

Neural Lineage Detection

- **Neural Lineage(CVPR 2024)**
  - **Learning based approach**
    - **실험 결과 1 (Learning rate와 iteration에 따른 차이)**
      - Learning rate와 iteration는 lineage detection accuracy에 큰 영향을 줌
      - 동일한 learning rate와 iteration이 진행되어야 됨



(a) Different learning rate (b) Different iteration

Figure 5. Lineage detection result when finetuning ResNet18 on CIFAR10 with different learning rates and iterations.

Neural Lineage Detection

- **Neural Lineage(CVPR 2024)**
  - **Learning based approach**
    - **실험 결과 2 (Finetuning generation 에 따른 차이**
      - G1: 7개의 ResNet 18 부모 모델으로 정의
      - 이 모델을 3세대에 걸쳐 finetuning 진행
        - ✓G2: EMNIST-Letters로 진행
        - ✓G3: FMNIST
        - ✓G4: EMNIST-Balanced
      - Learning based approach가 가장 우수
      - 세대 간격이 커질수록 검출 성능 감소
        - ✓세대 간격이 커질수록 검출 난이도 증가
        - ✓검출 정확도가 감소한다는 것을 실험을 통해 확인

		G2	G3	G4
G1	l_1 w/o Appx.	90.47±0.84	89.07±1.72	86.31±2.47
	CKA w/o Appx.	75.59±2.19	44.44±2.89	38.09±1.15
	l_1 w Appx.	98.92±0.59	97.02±0.51	96.83±0.82
	CKA w Appx.	77.38±1.93	51.59±2.98	51.19±1.27
	Lineage Detector	99.11±0.36	98.81±0.13	97.75±1.53
G2	l_1 w/o Appx.	-	91.27±1.89	83.93±2.67
	CKA w/o Appx.	-	65.88±2.63	67.86±2.03
	l_1 w Appx.	-	98.41±0.53	95.24±1.71
	CKA w Appx.	-	75.41±2.81	72.03±3.17
	Lineage Detector	-	99.61±0.83	98.38±0.02
G3	l_1 w/o Appx.	-	-	94.05±1.49
	CKA w/o Appx.	-	-	66.67±1.42
	l_1 w Appx.	-	-	97.02±0.75
	CKA w Appx.	-	-	72.03±1.91
	Lineage Detector	-	-	98.41±0.98

Table 2. Results of across-generational lineage detection.

Neural Lineage Detection

- **Neural Lineage(CVPR 2024)**
  - **Learning based approach**
    - **실험 결과 2 (다른 task에 대해 적용)**
      - **Elastic Weight Consolidation(EWC) 정규화**
        - ✓파라미터가 부모 모델의 파라미터에서 너
      - **KL-divergence(KLD) 정규화**
        - ✓Finetuning된 모델의 출력이 다른 teacher 관계를 약화시킬 수 있음

		Detection	Segmentation	Continual Learning	Knowledge Distillation
w/o Approx.	l_1	95.86±2.03	25.01±3.97	98.25±0.12	93.38±1.66
	l_2	91.22±2.26	22.91±2.11	99.74±0.38	92.92±1.59
	l_∞	78.58±4.01	33.33±4.68	97.81±0.54	70.65±1.21
	CKA	73.20±4.54	24.98±5.89	39.04±1.11	44.03±3.25
	DC	71.76±4.81	20.85±4.24	37.84±1.02	34.43±3.01
w Approx.	l_1	96.42±1.23	31.27±3.45	99.32±0.12	93.57±1.75
	l_2	92.85±1.91	25.01±1.52	**99.83±0.21**	94.03±1.23
	l_p	91.07±1.57	35.41±2.29	99.82±0.23	82.57±2.19
	LSE	96.23±1.83	22.93±2.31	99.11±0.38	93.21±1.71
	CKA	78.56±4.43	24.99±5.95	41.23±1.11	46.79±3.38
	DC	73.21±4.45	27.07±3.84	38.61±1.31	34.86±2.74
Lineage Detector		**99.28±0.72**	**37.56±4.04**	99.18±0.63	**99.07±0.07**

Table 3. Lineage detection results for other tasks and losses. The best (the second-best) ones are marked in **bold** (underlined).

Reference

- **Neural Lineage(CVPR 2024 oral)**
- **Neural Tangent Kernel: Convergence and Generalization in Neural Networks (NeurIPS 2018)**

감사합니다.