Average Gradient Outer Product:

A Mechanism for **Deep Neural Collapse**[1]

J. Setpal

October {10, 24}, 2024

[1]Beaglehole, Súkeník et. al. [NeurIPS 2024]

Outline

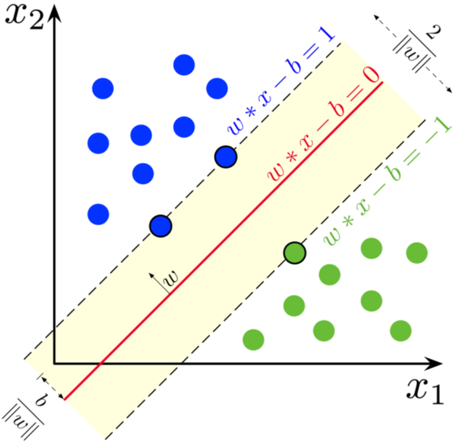
1. Background & Intuition
2. Deep Neural Collapse (DNC)
3. Average Gradient Outer Product (AGOP)
4. AGOP As a Mechanism for DNC

Outline

1. Background & Intuition
2. Deep Neural Collapse (DNC)
3. Average Gradient Outer Product (AGOP)
4. AGOP As a Mechanism for DNC

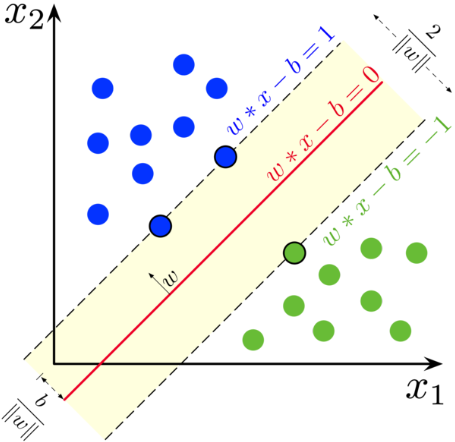
Primer – Support Vector Machines (SVMs)

We start with a linear SVM:

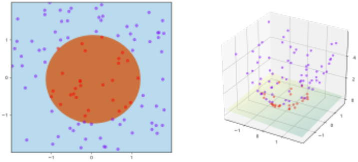


Primer – Support Vector Machines (SVMs)

We start with a linear SVM:

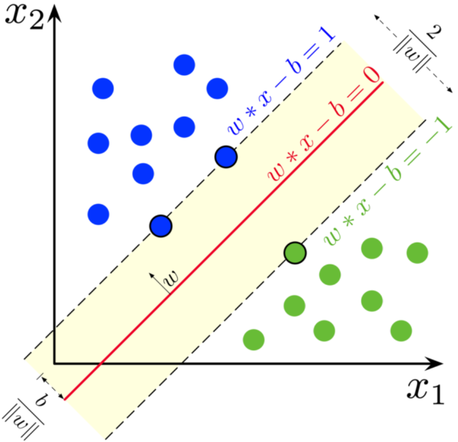


An approach to obtain a non-linear decision boundary is to learn a hyperplane in higher-dimensions:

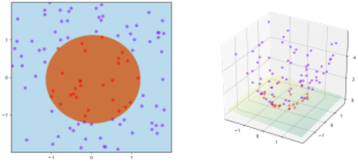


Primer – Support Vector Machines (SVMs)

We start with a linear SVM:



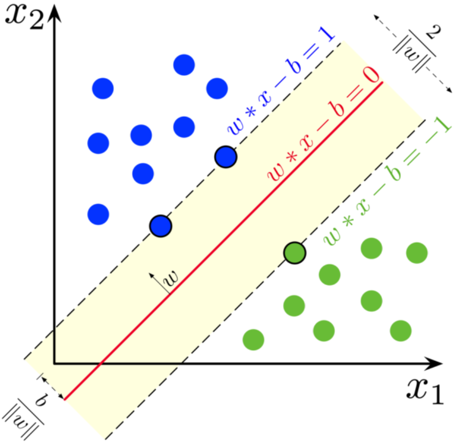
An approach to obtain a non-linear decision boundary is to learn a hyperplane in higher-dimensions:



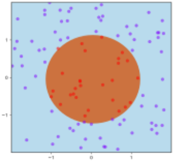
"Lazy" approaches to kernel choices include *polynomial* / *RBF* kernels.

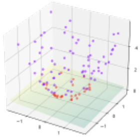
Primer – Support Vector Machines (SVMs)

We start with a linear SVM:



An approach to obtain a non-linear decision boundary is to learn a hyperplane in higher-dimensions:





“Lazy” approaches to kernel choices include *polynomial* / *RBF* kernels.

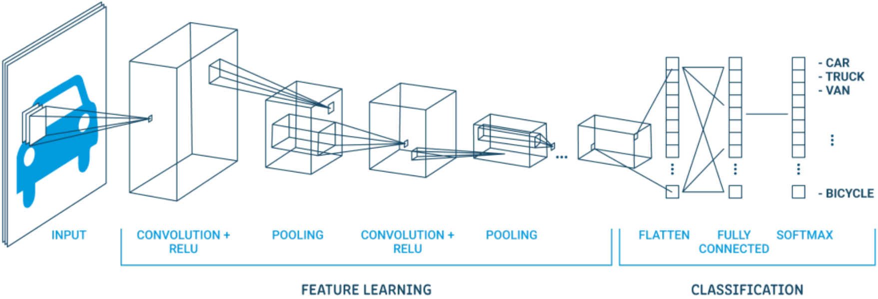
The “laziest” kernel of all is a **deep neural network**.

Neural Networks are Incredibly Overparameterized

Our study today is constrained to classifiers.

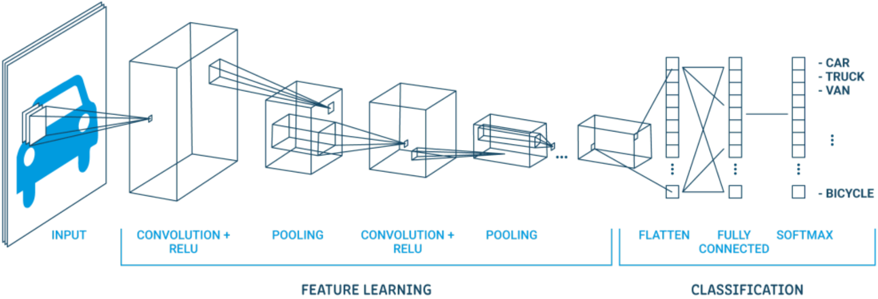
Neural Networks are Incredibly Overparameterized

Our study today is constrained to classifiers. WLOG, we can constrain our study to **image classifiers**.



Neural Networks are Incredibly Overparameterized

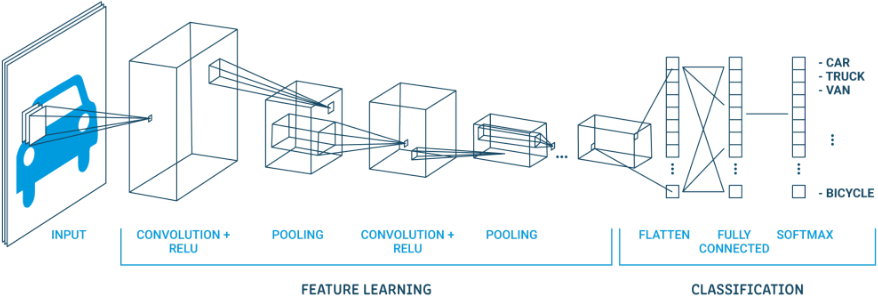
Our study today is constrained to classifiers. WLOG, we can constrain our study to **image classifiers**.



Traditional Learning: $n \geq d; \; W \in \mathbb{R}^d, \; \mathcal{D} = \{(x_i, y_i)\}_{i=1}^n$

Neural Networks are Incredibly Overparameterized

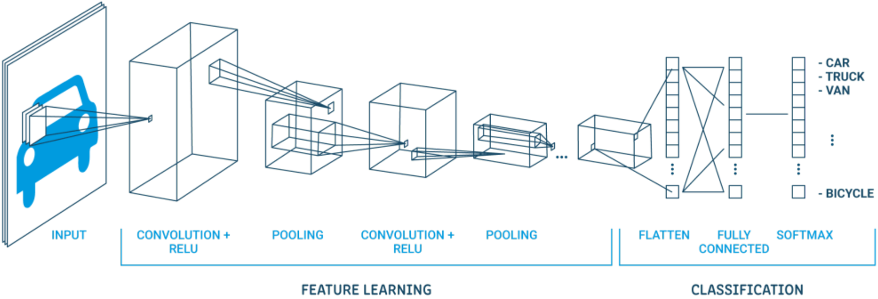
Our study today is constrained to classifiers. WLOG, we can constrain our study to **image classifiers**.



Traditional Learning: $n \geq d$; $W \in \mathbb{R}^d$, $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^n$
Overparameterized Learning: $d \geq n$

Neural Networks are Incredibly Overparameterized

Our study today is constrained to classifiers. WLOG, we can constrain our study to **image classifiers**.



Traditional Learning: $n \geq d$; $W \in \mathbb{R}^d$, $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^n$
Overparameterized Learning: $d \geq n$

Q: Why does overparameterized learning generalize?

Outline

1. Background & Intuition
2. Deep Neural Collapse (DNC)
3. Average Gradient Outer Product (AGOP)
4. AGOP As a Mechanism for DNC

What is Deep Neural Collapse (DNC)?

Deep Neural Collapse is a phenomenon describing *rigidity* in the feature representation(s) of the final layer(s) of *overtrained* Deep Neural Networks.

What is Deep Neural Collapse (DNC)?

Deep Neural Collapse is a phenomenon describing *rigidity* in the feature representation(s) of the final layer(s) of *overtrained* Deep Neural Networks.

$\mathbf{Q}_1$: What does *overtrained* mean?
$\mathbf{A}_1$: When a sufficiently expressive network h trained to minimize $\mathcal{L}(S_n)$ satisfies $h(x_i) = y_i \ \forall i$, it reaches the **Terminal Point of Training**. When trained beyond this point, the model is overtrained.

What is Deep Neural Collapse (DNC)?

Deep Neural Collapse is a phenomenon describing *rigidity* in the feature representation(s) of the final layer(s) of *overtrained* Deep Neural Networks.

$\mathbf{Q}_1$: What does *overtrained* mean?
$\mathbf{A}_1$: When a sufficiently expressive network h trained to minimize $\mathcal{L}(S_n)$ satisfies $h(x_i) = y_i \ \forall i$, it reaches the **Terminal Point of Training**. When trained beyond this point, the model is overtrained.

$\mathbf{Q}_2$: What does *rigidity* mean?
$\mathbf{A}_2$: We quantify *rigidity* by 4 key metrics, which iff satisifed, implies DNC.

What is Deep Neural Collapse (DNC)?

Deep Neural Collapse is a phenomenon describing *rigidity* in the feature representation(s) of the final layer(s) of *overtrained* Deep Neural Networks.

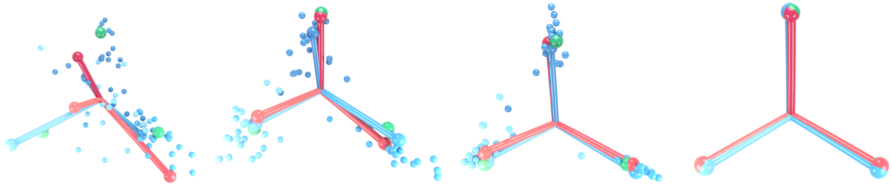
$\mathbf{Q}_1$: What does *overtrained* mean?
$\mathbf{A}_1$: When a sufficiently expressive network h trained to minimize $\mathcal{L}(S_n)$ satisfies $h(x_i) = y_i \ \forall i$, it reaches the **Terminal Point of Training**. When trained beyond this point, the model is <u>overtrained</u>.

$\mathbf{Q}_2$: What does *rigidity* mean?
$\mathbf{A}_2$: We quantify *rigidity* by <u>4 key metrics</u>, which iff satisifed, implies DNC.

$\mathbf{Q}_{2_a}$: What are the 4 key metrics?
$\mathbf{A}_{2_a}$: Exactly what we'll discuss next!

$NC1$ – Collapse of Variability (1/2)

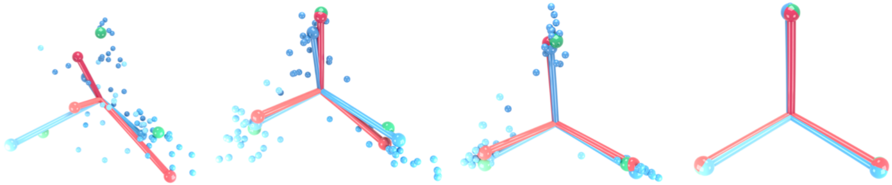
At a high level, the structure of the penultimate layer collapses towards:



Evolution of penultimate layer outputs on VGG13 trained on CIFAR10.

$NC1$ – Collapse of Variability (1/2)

At a high level, the structure of the penultimate layer collapses towards:

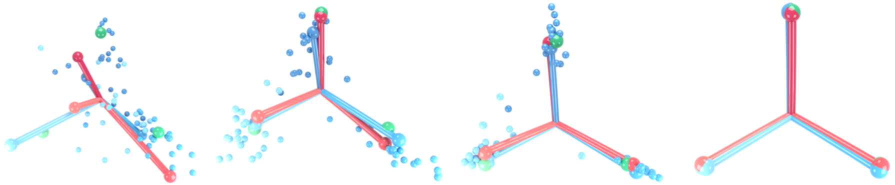


Evolution of penultimate layer outputs on VGG13 trained on CIFAR10.

For all classes $k \in [K]$, datapoints $i \in [n]$ within a class, & penultimate feature vector $f(k, i)$,

$NC1$ – Collapse of Variability (1/2)

At a high level, the structure of the penultimate layer collapses towards:



Evolution of penultimate layer outputs on VGG13 trained on CIFAR10.

For all classes $k \in [K]$, datapoints $i \in [n]$ <u>within a class</u>, & penultimate feature vector $f(k, i)$, we have class-specific & global means:

$$\boldsymbol{\mu}_k = \frac{1}{n} \sum_{i=1}^{n} f(k, i) \tag{1}$$

$$\boldsymbol{\mu}_G = \frac{1}{K} \sum_{i=1}^{K} \boldsymbol{\mu}_k \tag{2}$$

$NC1$ – Collapse of Variability (2/2)

We can use them to calculate *intra* and *inter*-class differences:

$$\mathrm{Cov}_W = \frac{1}{Kn}\sum_{k=1}^{K}\sum_{i=1}^{n}((f(k,i)-\boldsymbol{\mu}_k)(f(k,i)-\boldsymbol{\mu}_k)^T) \in \mathbb{R}^{m\times m} \qquad (3)$$

$$\mathrm{Cov}_B = \frac{1}{K}\sum_{k=1}^{K}((\boldsymbol{\mu}_k-\boldsymbol{\mu}_G)(\boldsymbol{\mu}_k-\boldsymbol{\mu}_G)^T) \in \mathbb{R}^{m\times m} \qquad (4)$$

$NC1$ – Collapse of Variability (2/2)

We can use them to calculate *intra* and *inter*-class differences:

$$\mathrm{Cov}_W = \frac{1}{Kn} \sum_{k=1}^{K} \sum_{i=1}^{n} ((f(k,i) - \boldsymbol{\mu}_k)(f(k,i) - \boldsymbol{\mu}_k)^T) \in \mathbb{R}^{m \times m} \tag{3}$$

$$\mathrm{Cov}_B = \frac{1}{K} \sum_{k=1}^{K} ((\boldsymbol{\mu}_k - \boldsymbol{\mu}_G)(\boldsymbol{\mu}_k - \boldsymbol{\mu}_G)^T) \in \mathbb{R}^{m \times m} \tag{4}$$

Which we combine to measure overall **variability collapse**:

$$NC1 := \frac{1}{K} \mathrm{Tr}\left(\mathrm{Cov}_W \, \mathrm{Cov}_B^{\dagger}\right) \tag{5}$$

Aside: Psuedoinverses

The **inverse** of a matrix A is defined s.t. it satisfies the following condition:

$$A, B, I \in \mathbb{R}^{d \times d} \text{ s.t. } AB = BA = I_d; \; B := A^{-1}, \; A := B^{-1} \tag{6}$$

Aside: Psuedoinverses

The **inverse** of a matrix A is defined s.t. it satisfies the following condition:

$$A, B, I \in \mathbb{R}^{d \times d} \text{ s.t. } AB = BA = I_d;\ B := A^{-1},\ A := B^{-1} \tag{6}$$

What about when $X \in \mathbb{R}^{n \times m}$?

Aside: Psuedoinverses

The **inverse** of a matrix A is defined s.t. it satisfies the following condition:

$$A, B, I \in \mathbb{R}^{d\times d} \text{ s.t. } AB = BA = I_d;\ B := A^{-1},\ A := B^{-1} \tag{6}$$

What about when $X \in \mathbb{R}^{n\times m}$? A **psuedoinverse** is a *generalized inverse*, which instead satisfies the following four conditions:

$$XX^{-1}X = X \tag{7}$$

$$X^{-1}XX^{-1} = X^{-1} \tag{8}$$

$$(XX^{-1})^{*} = XX^{-1} \tag{9}$$

$$X^{-1}X^{*} = X^{-1}X \tag{10}$$

Where X^{*} is the conjugate transpose of X.

Aside: Psuedoinverses

The **inverse** of a matrix A is defined s.t. it satisfies the following condition:

$$A, B, I \in \mathbb{R}^{d \times d} \text{ s.t. } AB = BA = I_d;\ B := A^{-1},\ A := B^{-1} \tag{6}$$

What about when $X \in \mathbb{R}^{n \times m}$? A **psuedoinverse** is a *generalized inverse*, which instead satisfies the following four conditions:

$$XX^{-1}X = X \tag{7}$$

$$X^{-1}XX^{-1} = X^{-1} \tag{8}$$

$$(XX^{-1})^* = XX^{-1} \tag{9}$$

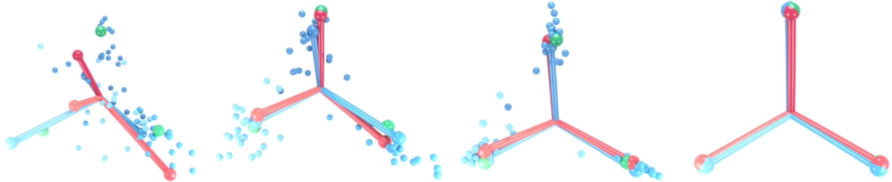
$$X^{-1}X^* = X^{-1}X \tag{10}$$

Where X^* is the conjugate transpose of X.

Implication: We can compute correlation b/w general matrix dimensions.

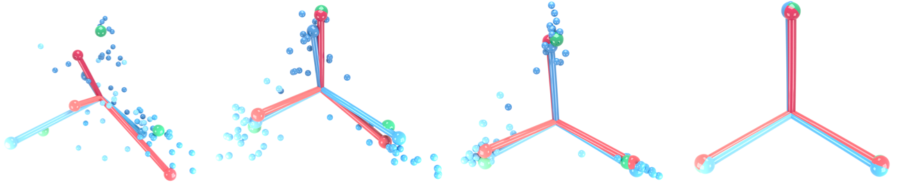
$NC2$ – The Simplex ETF (1/2)

This time, we can on focus the **structure** of the class means:



*NC*2 – The Simplex ETF (1/2)

This time, we can on focus the **structure** of the class means:

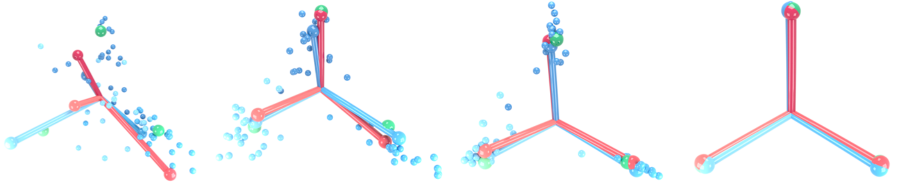


A useful analogy is *VSEPR*[2] from Chemistry.

[2] I sincerely apologize for making this reference.

*NC*2 – The Simplex ETF (1/2)

This time, we can on focus the **structure** of the class means:

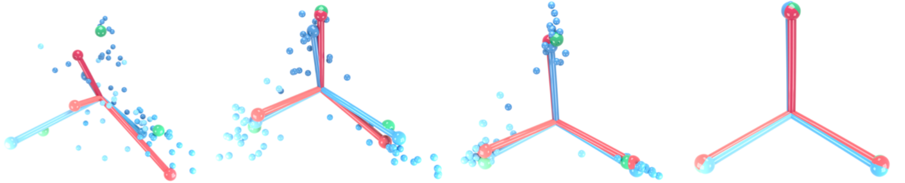


A useful analogy is *VSEPR*[2] from Chemistry. Each class (atom) repels the other creating a **simplex equiangular tight frame** (simplex ETF).

[2] I sincerely apologize for making this reference.

$NC2$ – The Simplex ETF (1/2)

This time, we can on focus the **structure** of the class means:



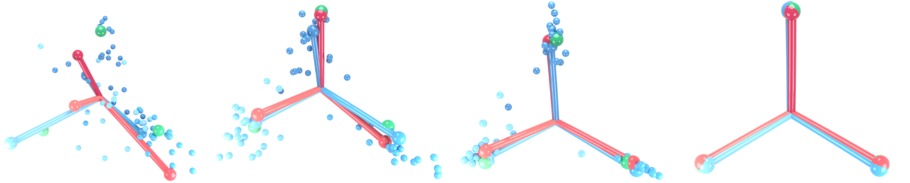
A useful analogy is *VSEPR*[2] from Chemistry. Each class (atom) repels the other creating a **simplex equiangular tight frame** (simplex ETF).

- **Simplex** is the simplest polytope (object with flat sides).

[2] I sincerely apologize for making this reference.

$NC2$ – The Simplex ETF (1/2)

This time, we can on focus the **structure** of the class means:



A useful analogy is *VSEPR*[2] from Chemistry. Each class (atom) repels the other creating a **simplex equiangular tight frame** (simplex ETF).

- **Simplex** is the simplest polytope (object with flat sides).
- **Equiangular Tight Frame** is a matrix $M \in \mathbb{R}^{K \times m}$ s.t.

$$|\langle \boldsymbol{m}_j, \boldsymbol{m}_k \rangle| = \alpha \ \exists \alpha \geq 0 \ \forall j, k \text{ s.t. } j \neq k \tag{11}$$

$$MM^T = \sqrt{\frac{C}{C-1}} \left(I_C - \frac{1}{C} \mathbb{1}_{C \times C} \right) \tag{12}$$

Satisfying equiangular and tight respectively.

[2] I sincerely apologize for making this reference.

$NC2$ – The Simplex ETF (2/2)

We can use this to define $NC2$. Given re-centered class means $\{\boldsymbol{\mu}_k - \boldsymbol{\mu}_G\}_{k \in [K]}$, they are **equidistant** if:

$$\|\boldsymbol{\mu}_k - \boldsymbol{\mu}_G\|_2 = \|\boldsymbol{\mu}_{k'} - \boldsymbol{\mu}_G\|_2 \ \forall k, k' \in [K] \tag{13}$$

$NC2$ – The Simplex ETF (2/2)

We can use this to define $NC2$. Given re-centered class means $\{\boldsymbol{\mu}_k - \boldsymbol{\mu}_G\}_{k\in[K]}$, they are **equidistant** if:

$$\|\boldsymbol{\mu}_k - \boldsymbol{\mu}_G\|_2 = \|\boldsymbol{\mu}_{k'} - \boldsymbol{\mu}_G\|_2 \ \forall k, k' \in [K] \tag{13}$$

We then normalize each feature vector to create our simplex ETF:

$$M = \text{Concat}\left(\left\{\frac{\boldsymbol{\mu}_k - \boldsymbol{\mu}_G}{\|\boldsymbol{\mu}_k - \boldsymbol{\mu}_G\|_2} \in \mathbb{R}^m\right\}^{[K]}\right) \in \mathbb{R}^{K\times m} \tag{14}$$

$NC2$ – The Simplex ETF (2/2)

We can use this to define $NC2$. Given re-centered class means $\{\boldsymbol{\mu}_k - \boldsymbol{\mu}_G\}_{k\in[K]}$, they are **equidistant** if:

$$\|\boldsymbol{\mu}_k - \boldsymbol{\mu}_G\|_2 = \|\boldsymbol{\mu}_{k'} - \boldsymbol{\mu}_G\|_2 \; \forall k, k' \in [K] \tag{13}$$

We then normalize each feature vector to create our simplex ETF:

$$M = \text{Concat}\left(\left\{\frac{\boldsymbol{\mu}_k - \boldsymbol{\mu}_G}{\|\boldsymbol{\mu}_k - \boldsymbol{\mu}_G\|_2} \in \mathbb{R}^m\right\}^{[K]}\right) \in \mathbb{R}^{K\times m} \tag{14}$$

M is now compared to it's distance from the simplex ETF:

$$NC2 := \left\| \underbrace{\frac{MM^T}{\|MM^T\|_F}}_{\text{feature vector as a simplex}} - \underbrace{\frac{1}{\sqrt{K-1}}\left(I_K - \frac{\mathbb{1}_{K\times K}}{K}\right)}_{\text{canonical simplex}} \right\|_F \tag{15}$$

Setting up our second metric.

$NC3$ – Self-Dual Alignment

The final layer's weights $W \in \mathbb{R}^{K \times m}$ align with simplex ETF of M:

$$\frac{A}{\|A\|_F} \propto \frac{M}{\|M\|_F} \tag{16}$$

$NC3$ – Self-Dual Alignment

The final layer's weights $W \in \mathbb{R}^{K \times m}$ align with simplex ETF of M:

$$\frac{A}{\|A\|_F} \propto \frac{M}{\|M\|_F} \tag{16}$$

We can use this to setup the third metric:

$$NC3 := \left\| \underbrace{\frac{AM^T}{\|AM^T\|_F}}_{\equiv \text{ cosine similiarity}} - \underbrace{\frac{1}{\sqrt{K-1}}\left(I_K - \frac{\mathbb{1}_{K\times K}}{K}\right)}_{\text{canonical simplex}} \right\|_F \tag{17}$$

$NC4$ – Secretly k-NN

Finally, we observe that for x_{n+1}, the classification result $\equiv$ k-NN rule:

$$\arg\max \hat{y}_{n+1} = \operatorname*{arg\,min}_{k \in [K]} \|f(x_{n+1}) - \boldsymbol{\mu}_k\|_2 \tag{18}$$

$NC4$ – Secretly k-NN

Finally, we observe that for x_{n+1}, the classification result $\equiv$ k-NN rule:

$$\arg\max \hat{y}_{n+1} = \arg\min_{k \in [K]} \|f(x_{n+1}) - \boldsymbol{\mu}_k\|_2 \tag{18}$$

Which we can use to setup our final metric:

$$NC4 : \frac{1}{Kn} \sum_{k=1}^{K} \sum_{i=1}^{n} \mathbb{1} \left[\arg\max \hat{y}_i \neq \arg\min_{k \in [K]} \|f(x_i) - \boldsymbol{\mu}_k\|_2 \right] \tag{19}$$

$NC4$ – Secretly k-NN

Finally, we observe that for x_{n+1}, the classification result $\equiv$ k-NN rule:

$$\arg\max \hat{y}_{n+1} = \arg\min_{k \in [K]} \|f(x_{n+1}) - \boldsymbol{\mu}_k\|_2 \tag{18}$$

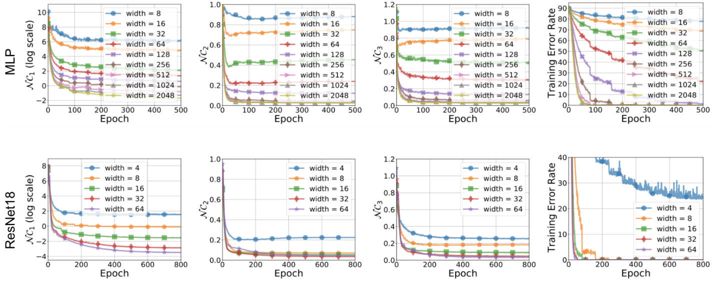
Which we can use to setup our final metric:

$$NC4 : \frac{1}{Kn} \sum_{k=1}^{K} \sum_{i=1}^{n} \mathbb{1} \left[\arg\max \hat{y}_i \neq \arg\min_{k \in [K]} \|f(x_i) - \boldsymbol{\mu}_k\|_2 \right] \tag{19}$$

If each of the 4 previous metrics $\to 0$, the network is considered **collapsed**.

Do We Even Want This? – Data Independence

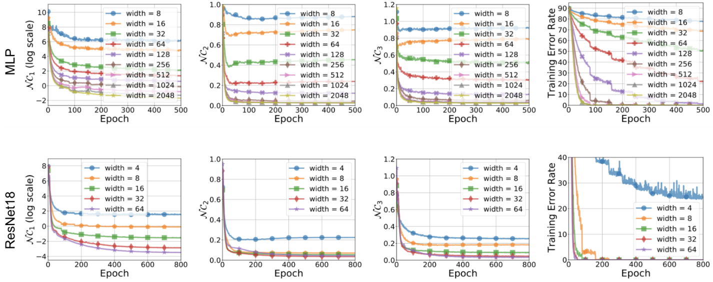
Here's what the metric convergence plots look like, with *random labels*.



(a) NC1 (b) NC2 (c) NC3 (d) Training error

Do We Even Want This? – Data Independence

Here's what the metric convergence plots look like, with *random labels*.

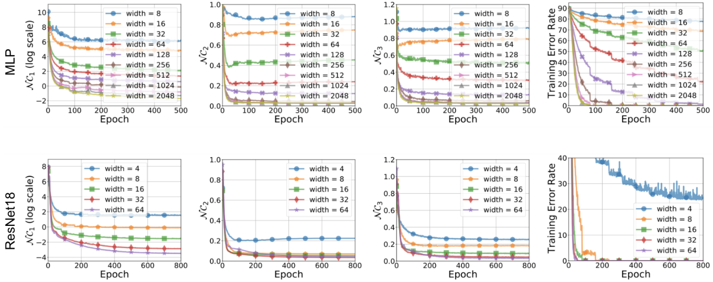


(a) NC1 (b) NC2 (c) NC3 (d) Training error

Q: Do we even want this?

Do We Even Want This? – Data Independence

Here's what the metric convergence plots look like, with *random labels*.



(a) NC1 (b) NC2 (c) NC3 (d) Training error

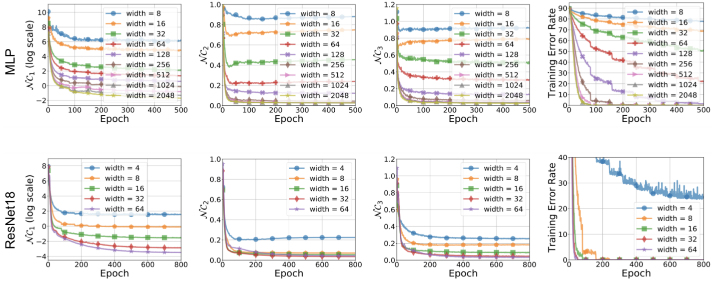
Q: Do we even want this?

A: Yes. Here's some reasons why:

1. **OOD Inference**: If we have a point outside the simplex ETF, it is likely outside the training distribution.

Do We Even Want This? – Data Independence

Here's what the metric convergence plots look like, with *random labels*.



(a) NC1 (b) NC2 (c) NC3 (d) Training error

Q: Do we even want this?

A: Yes. Here's some reasons why:

1. **OOD Inference**: If we have a point outside the simplex ETF, it is likely outside the training distribution.
2. **Forced ETF**: The final layer can be a fixed as a simplex!

Outline

1 Background & Intuition

2 Deep Neural Collapse (DNC)

3 Average Gradient Outer Product (AGOP)

4 AGOP As a Mechanism for DNC

What is AGOP?

Average Gradient Outer Product (AGOP) is a data-dependent, backpropagation-free mechanism that **characterizes feature learning in neural networks**.

What is AGOP?

Average Gradient Outer Product (AGOP) is a data-dependent, backpropagation-free mechanism that **characterizes feature learning in neural networks**.

Given dataset $X \in \mathbb{R}^{d_0 \times N} \sim \mathcal{D}^n$, $f : \mathbb{R}^{d_0 \times 1} \to \mathbb{R}^{K \times 1}$, we define AGOP:

$$\mathrm{AGOP}(f, X) := \underbrace{\frac{1}{N} \sum_{c=1}^{K} \sum_{i=1}^{N}}_{\text{average}} \underbrace{\underbrace{\frac{\partial f(x_{ci})}{\partial x_{ci}}}_{\text{gradient}} \frac{\partial f(x_{ci})}{\partial x_{ci}}^{T}}_{\text{outer product}} \tag{20}$$

What is AGOP?

Average Gradient Outer Product (AGOP) is a data-dependent, backpropagation-free mechanism that **characterizes feature learning in neural networks**.

Given dataset $X \in \mathbb{R}^{d_0 \times N} \sim \mathcal{D}^n$, $f : \mathbb{R}^{d_0 \times 1} \to \mathbb{R}^{K \times 1}$, we define AGOP:

$$\mathrm{AGOP}(f, X) := \underbrace{\frac{1}{N} \sum_{c=1}^{K} \sum_{i=1}^{N}}_{\text{average}} \underbrace{\underbrace{\frac{\partial f(x_{ci})}{\partial x_{ci}}}_{\text{gradient}} \frac{\partial f(x_{ci})}{\partial x_{ci}}^{T}}_{\text{outer product}} \tag{20}$$

Why is this useful?

What is AGOP?

Average Gradient Outer Product (AGOP) is a data-dependent, backpropagation-free mechanism that **characterizes feature learning in neural networks**.

Given dataset $X \in \mathbb{R}^{d_0 \times N} \sim \mathcal{D}^n$, $f : \mathbb{R}^{d_0 \times 1} \to \mathbb{R}^{K \times 1}$, we define AGOP:

$$\mathrm{AGOP}(f, X) := \underbrace{\frac{1}{N} \sum_{c=1}^{K} \sum_{i=1}^{N}}_{\text{average}} \underbrace{\underbrace{\frac{\partial f(x_{ci})}{\partial x_{ci}}}_{\text{gradient}} \frac{\partial f(x_{ci})}{\partial x_{ci}}^{T}}_{\text{outer product}} \tag{20}$$

Why is this useful? $\mathrm{AGOP}(\hat{f}, X) \approx \mathrm{EGOP}(f^*, \mathcal{D})$:

$$\mathrm{EGOP}(f^*, \mathcal{D}) := \mathbb{E}_{\mathcal{D}} \left[\frac{\partial f^*(x_{ci})}{\partial x_{ci}} \frac{\partial f^*(x_{ci})}{\partial x_{ci}}^{T} \right] \tag{21}$$

What is AGOP?

Average Gradient Outer Product (AGOP) is a data-dependent, backpropagation-free mechanism that **characterizes feature learning in neural networks**.

Given dataset $X \in \mathbb{R}^{d_0 \times N} \sim \mathcal{D}^n$, $f : \mathbb{R}^{d_0 \times 1} \to \mathbb{R}^{K \times 1}$, we define AGOP:

$$\mathrm{AGOP}(f, X) := \underbrace{\frac{1}{N} \sum_{c=1}^{K} \sum_{i=1}^{N}}_{\text{average}} \underbrace{\underbrace{\frac{\partial f(x_{ci})}{\partial x_{ci}}}_{\text{gradient}} \frac{\partial f(x_{ci})}{\partial x_{ci}}^T}_{\text{outer product}} \tag{20}$$

Why is this useful? $\mathrm{AGOP}(\hat{f}, X) \approx \mathrm{EGOP}(f^*, \mathcal{D})$:

$$\mathrm{EGOP}(f^*, \mathcal{D}) := \mathbb{E}_{\mathcal{D}} \left[\frac{\partial f^*(x_{ci})}{\partial x_{ci}} \frac{\partial f^*(x_{ci})}{\partial x_{ci}}^T \right] \tag{21}$$

EGOP contains useful information like low-rank structure, that can improves predictions.

Neural Feature Matrices (NFMs) & Ansatz (NFA)

Neural Feature Matrices (NFMs) are right singular $\{$vectors, values$\}$ of $\{W_i^{(l)}\}_{i=1}^{k}$ – these rotate, sale, reflect the input.

Neural Feature Matrices (NFMs) & Ansatz (NFA)

Neural Feature Matrices (NFMs) are right singular $\{$vectors, values$\}$ of $\{W_i^{(l)}\}_{i=1}^{k}$ – these rotate, sale, reflect the input.

Singular $\{$vectors, values$\}$ may be recovered from eigen $\{$vectors, values$\}$ of $W^T W$.

Neural Feature Matrices (NFMs) & Ansatz (NFA)

Neural Feature Matrices (NFMs) are right singular {vectors, values} of $\{W_i^{(l)}\}_{i=1}^{k}$ – these rotate, sale, reflect the input.

Singular {vectors, values} may be recovered from eigen {vectors, values} of $W^T W$. The vectors capture **task relevant directions** used for **identifying features**:

$$NFM(W_l) = {W_l}^T W_l = S_L \Sigma^2 S_R^T \ \forall l \in [L] \tag{22}$$

Neural Feature Matrices (NFMs) & Ansatz (NFA)

Neural Feature Matrices (NFMs) are right singular {vectors, values} of $\{W_i^{(l)}\}_{i=1}^{k}$ – these rotate, sale, reflect the input.

Singular {vectors, values} may be recovered from eigen {vectors, values} of $W^T W$. The vectors capture **task relevant directions** used for **identifying features**:

$$NFM(W_l) = W_l^T W_l = S_L \Sigma^2 S_R^T \ \forall l \in [L] \tag{22}$$

These are defined for each intermediate layer.

Neural Feature Matrices (NFMs) & Ansatz (NFA)

Neural Feature Matrices (NFMs) are right singular {vectors, values} of $\{W_i^{(l)}\}_{i=1}^{k}$ – these rotate, sale, reflect the input.

Singular {vectors, values} may be recovered from eigen {vectors, values} of $W^T W$. The vectors capture **task relevant directions** used for **identifying features**:

$$NFM(W_l) = W_l^T W_l = S_L \Sigma^2 S_R^T \ \forall l \in [L] \tag{22}$$

These are defined for each intermediate layer. Each NFM is connected to the AGOP of it's layer:

$$\rho\left(W_l^T W_l,\ \frac{1}{N}\sum_{c=1}^{K}\sum_{i=1}^{N}\frac{\partial f(x_{ci})}{\partial f(x_{ci})_l}\frac{\partial f(x_{ci})}{\partial f(x_{ci})_l}\right) \approx 1 \tag{23}$$

This is called the **Neural Feature Ansatz (NFA)**.

Neural Feature Matrices (NFMs) & Ansatz (NFA)

Neural Feature Matrices (NFMs) are right singular $\{$vectors, values$\}$ of $\{W_i^{(l)}\}_{i=1}^{k}$ – these rotate, sale, reflect the input.

Singular $\{$vectors, values$\}$ may be recovered from eigen $\{$vectors, values$\}$ of $W^T W$. The vectors capture **task relevant directions** used for **identifying features**:

$$NFM(W_l) = W_l^T W_l = S_L \Sigma^2 S_R^T \;\forall l \in [L] \tag{22}$$

These are defined for each intermediate layer. Each NFM is connected to the AGOP of it's layer:

$$\rho\left(W_l^T W_l,\; \frac{1}{N}\sum_{c=1}^{K}\sum_{i=1}^{N} \frac{\partial f(x_{ci})}{\partial f(x_{ci})_l}\frac{\partial f(x_{ci})}{\partial f(x_{ci})_l}\right) \approx 1 \tag{23}$$

This is called the **Neural Feature Ansatz (NFA)**.
Bonus: This makes AGOP backpropogation-free.

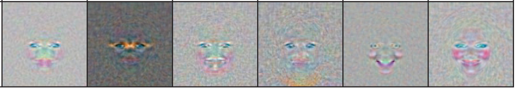
Features Identified by AGOP

So, what *exactly* does AGOP find?

Features Identified by AGOP

So, what *exactly* does AGOP find? **A lot**.

A Features captured by AGOP of Laplace kernel trained on CelebA prediction tasks

Task	Lipstick	Glasses	5 o'clock shadow	Necktie	Smiling	Rosy Cheeks
AGOP of Laplace Kernel (Top Eigenvector)						

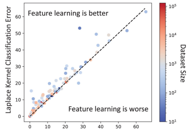
B

	Classification Tasks (Test Accuracy)							Regression Tasks (Test R^2)	
	CelebA Lipstick	CelebA Glasses	CelebA 5 o'clock shadow	CelebA Necktie	CelebA Smiling	CelebA Rosy Cheeks	SVHN	Low Rank Polynomial (Vyas et al. 2022)	Low Rank Polynomial (Damian et al. 2022)
Laplace Kernel + AGOP	**91.62%**	**94.06%**	**88.18%**	**90.39%**	**91.24%**	**88.72%**	**83.25%**	**0.997**	**0.941**
Laplace Kernel	90.89%	90.19%	84.90%	88.92%	90.00%	86.52%	74.83%	0.495	0.481

C Performance across 121 classification tasks from Fernández-Delgado et al. 2014

Classifier	Avg. Test Accuracy
Laplace Kernel + AGOP	**85.39%**
Laplace Kernel	83.76%

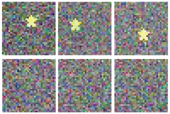
Laplace Kernel vs. Laplace Kernel + AGOP



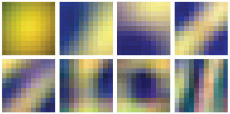
D Data Samples

Class 1

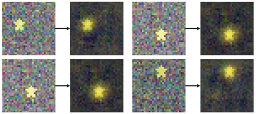
Class 0



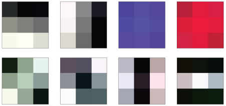
E Top 8 Eigenvectors of CNTK AGOP



F Transforming data samples with CNTK AGOP

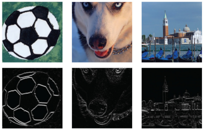


G Eigenvectors of CNTK AGOP (SVHN)



H ImageNet Images

+CNTK AGOP



I Performance Comparison

Dataset	CNTK Test Acc. (%)	CNTK +AGOP Test Acc. (%)
Stars in Noise	80.20	**99.70**
MNIST in Noise	72.55	**79.03**
SVHN	81.38	**85.02**
CIFAR-10	67.74	**67.97**
CIFAR-100	**37.64**	**37.64**
GTSRB	91.76	**93.02**
EMNIST	86.01	**86.73**

Outline

1. Background & Intuition
2. Deep Neural Collapse (DNC)
3. Average Gradient Outer Product (AGOP)
4. AGOP As a Mechanism for DNC

AGOP As a Mechanism for DNC (1/2)

Observation: *within*-class variability collapse occurs predominantly through multiplication by right singular-structure of weights (NFMs).

AGOP As a Mechanism for DNC (1/2)

Observation: *within*-class variability collapse occurs predominantly through multiplication by right singular-structure of weights (NFMs).

We first decompose W_l using SVD:

$$W_l = U_l S_l V_l^T \tag{24}$$

AGOP As a Mechanism for DNC (1/2)

Observation: *within*-class variability collapse occurs predominantly through multiplication by right singular-structure of weights (NFMs).

We first decompose W_l using SVD:

$$W_l = U_l S_l V_l^T \tag{24}$$

Then we view $S_l V_l^T$ as the affine input transformation, & $\sigma \circ U_l$ as applying elementwise non-linearity.

AGOP As a Mechanism for DNC (1/2)

Observation: *within*-class variability collapse occurs predominantly through multiplication by right singular-structure of weights (NFMs).

We first decompose W_l using SVD:

$$W_l = U_l S_l {V_l}^T \tag{24}$$

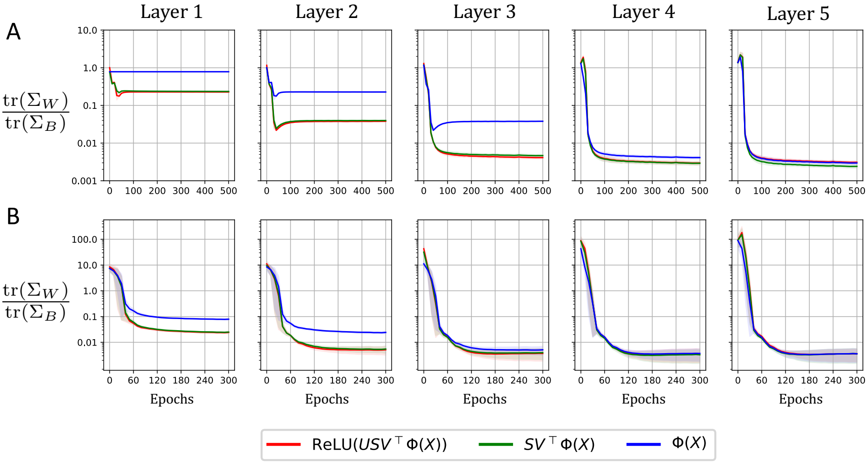
Then we view $S_l {V_l}^T$ as the affine input transformation, & $\sigma \circ U_l$ as applying elementwise non-linearity.

We see that computing $NC1(S_l {V_l}^T) \equiv NC1(W_l)$:

$$\begin{aligned} &\frac{1}{K} \operatorname{Tr}\left(\operatorname{Cov}_W(W_l) \operatorname{Cov}_B^{\dagger}(W_l)\right) \\ \equiv\ &\frac{1}{K} \operatorname{Tr}\left(\operatorname{Cov}_W(U_l S_l {V_l}^T U_l^T) \operatorname{Cov}_B^{\dagger}(U_l S_l {V_l}^T U_l^T)\right) \\ \equiv\ &\frac{1}{K} \operatorname{Tr}\left(\operatorname{Cov}_W(S_l {V_l}^T) \operatorname{Cov}_B^{\dagger}(S_l {V_l}^T)\right) \end{aligned} \tag{25}$$

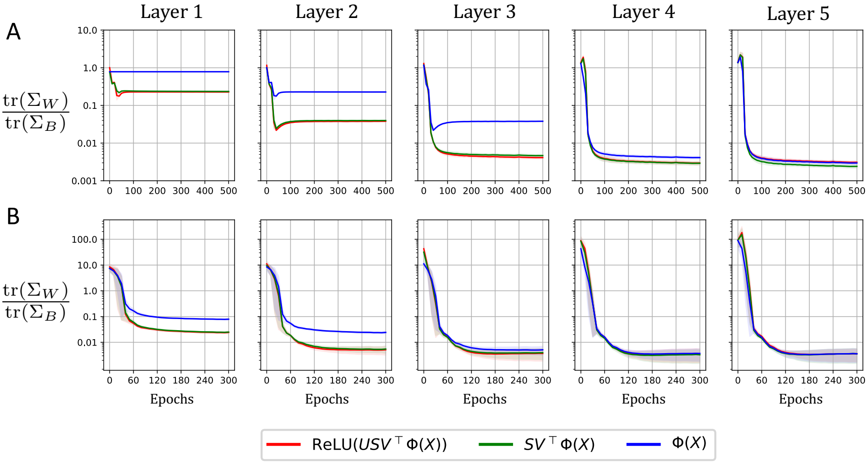
AGOP As a Mechanism for DNC (2/2)

We can identify the collapse occurs by the right-singular structure:



AGOP As a Mechanism for DNC (2/2)

We can identify the collapse occurs by the right-singular structure:



Additionally, $NC2$ (Simplex ETF) was observed, but only in the last layer.

Thank you!

Have an awesome rest of your day!

Slides: `https://cs.purdue.edu/homes/jsetpal/slides/dnc_by_agop.pdf`