Project Update: Transformers on Graph Connectivity

Qilin Ye, Deqing Fu, Robin Jia, and Vatsal Sharan

March 7, 2024

Problem Setup - Generating Graphs

(∗) Erdös-Renyi

(∗) Given n nodes, find $p(n) \in (0, 1)$ such that
$\mathbb{P}(\text{two nodes are disconnected}) \approx 0.5$ (Δ)

† For $n = 50$, $p(n) \approx 0.0355$.

‡ (Δ) implies $\lim_n np(n) > 1$ so w.h.p. there exists a *giant component* being the only one with $\geqslant O(\log n)$ vertices.

Problem Setup - Generating Graphs

(∗) Erdös-Renyi

(∗) Given n nodes, find $p(n) \in (0, 1)$ such that
$\mathbb{P}(\text{two nodes are disconnected}) \approx 0.5$ (Δ)

† For $n = 50$, $p(n) \approx 0.0355$.

‡ (Δ) implies $\lim_n np(n) > 1$ so w.h.p. there exists a *giant component* being the only one with $\geqslant O(\log n)$ vertices.

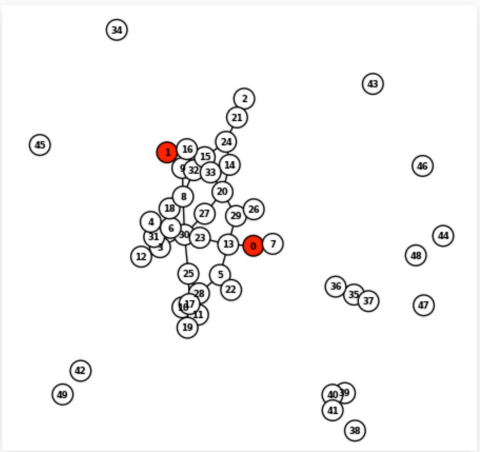
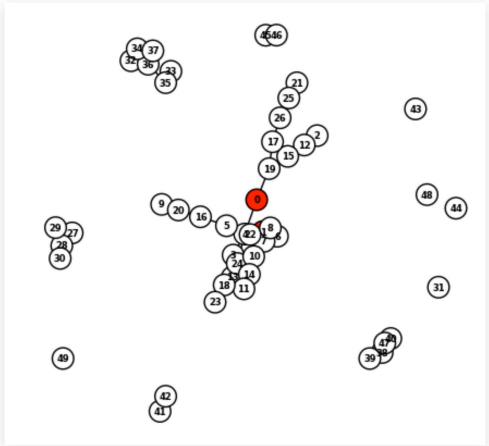
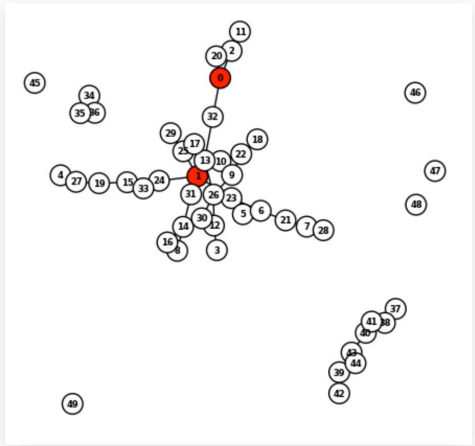
Problem Setup - Generating Graphs

(∗) Erdös-Renyi

(∗) Given n nodes, find $p(n) \in (0, 1)$ such that
$\mathbb{P}(\text{two nodes are disconnected}) \approx 0.5$ (Δ)

† For $n = 50$, $p(n) \approx 0.0355$.

‡ (Δ) implies $\lim_n np(n) > 1$ so w.h.p. there exists a *giant component* being the only one with $\geqslant O(\log n)$ vertices.

Problem Setup - Model

- (∗) `GPT2ModelFullAttentionForGraph`
- (∗) **Learnable embeddings** w/ dimension $n_{\text{dim}} \times n_{\text{nodes}} = 256 \times 50$.
- (∗) **Full attention**; no casual masking.
- (∗) ‘ **Identity positional embeddings**: self loops.
- (∗) Input: $[(\text{one-hot query1})^T \mid (\text{adj matrix}) \mid (\text{one-hot query2})^T]^T$
  - † Dimension: $(n_{\text{nodes}} + 2) \times n_{\text{nodes}}$
  - † Queries unused until prediction
- (∗) Two readout schemes:
  - † `inner_product`: ‘yes’ $\Leftrightarrow \langle \texttt{embd\_1}, \texttt{embd\_2} \rangle > 0$.
  - † `outer_product`: one *long* MLP:
    `[embd_1 | embd_2 | (embd_1 ⊗ embd_2).flatten()]`
- (∗) Typical training stuff... `AdamW`, weight decay, etc.

Problem Setup - Model

- (∗) `GPT2ModelFullAttentionForGraph`
- (∗) **Learnable embeddings** w/ dimension $n_{\text{dim}} \times n_{\text{nodes}} = 256 \times 50$.
- (∗) **Full attention**; no casual masking.
- (∗) ‘ **Identity positional embeddings**: self loops.
- (∗) Input: $[(\text{one-hot query1})^T \mid (\text{adj matrix}) \mid (\text{one-hot query2})^T]^T$
  - † Dimension: $(n_{\text{nodes}} + 2) \times n_{\text{nodes}}$
  - † Queries unused until prediction
- (∗) Two readout schemes:
  - † `inner_product`: ‘yes’ $\Leftrightarrow \langle \texttt{embd\_1}, \texttt{embd\_2} \rangle > 0$.
  - † `outer_product`: one *long* MLP:
    `[embd_1 | embd_2 | (embd_1 ⊗ embd_2).flatten()]`
- (∗) Typical training stuff... `AdamW`, weight decay, etc.

Problem Setup - Model

- (∗) `GPT2ModelFullAttentionForGraph`
- (∗) **Learnable embeddings** w/ dimension $n_{\text{dim}} \times n_{\text{nodes}} = 256 \times 50$.
- (∗) **Full attention**; no casual masking.
- (∗) ‘ **Identity positional embeddings**: self loops.
- (∗) Input: $[(\text{one-hot query1})^T \mid (\text{adj matrix}) \mid (\text{one-hot query2})^T]^T$
  - † Dimension: $(n_{\text{nodes}} + 2) \times n_{\text{nodes}}$
  - † Queries unused until prediction
- (∗) Two readout schemes:
  - † `inner_product`: ‘yes’ $\Leftrightarrow \langle$`embd_1, embd_2`$\rangle > 0$.
  - † `outer_product`: one *long* MLP:
    `[embd_1 | embd_2 | (embd_1` $\otimes$ `embd_2).flatten()]`
- (∗) Typical training stuff... `AdamW`, weight decay, etc.

Problem Setup - Model

- (∗) `GPT2ModelFullAttentionForGraph`
- (∗) **Learnable embeddings** w/ dimension $n_{\text{dim}} \times n_{\text{nodes}} = 256 \times 50$.
- (∗) **Full attention**; no casual masking.
- (∗) ‘ **Identity positional embeddings**: self loops.
- (∗) Input: $[(\text{one-hot query1})^T \mid (\text{adj matrix}) \mid (\text{one-hot query2})^T]^T$
  - † Dimension: $(n_{\text{nodes}} + 2) \times n_{\text{nodes}}$
  - † Queries unused until prediction
- (∗) Two readout schemes:
  - † `inner_product`: ‘yes’ $\Leftrightarrow \langle \texttt{embd\_1}, \texttt{embd\_2} \rangle > 0$.
  - † `outer_product`: one *long* MLP:
    `[embd_1 | embd_2 | (embd_1 ⊗ embd_2).flatten()]`
- (∗) Typical training stuff... `AdamW`, weight decay, etc.

Problem Setup - Model

(∗) `GPT2ModelFullAttentionForGraph`

(∗) **Learnable embeddings** w/ dimension $n_{\text{dim}} \times n_{\text{nodes}} = 256 \times 50$.

(∗) **Full attention**; no casual masking.

(∗) ‘ **Identity positional embeddings**: self loops.

(∗) Input: $[(\text{one-hot query1})^T \mid (\text{adj matrix}) \mid (\text{one-hot query2})^T]^T$

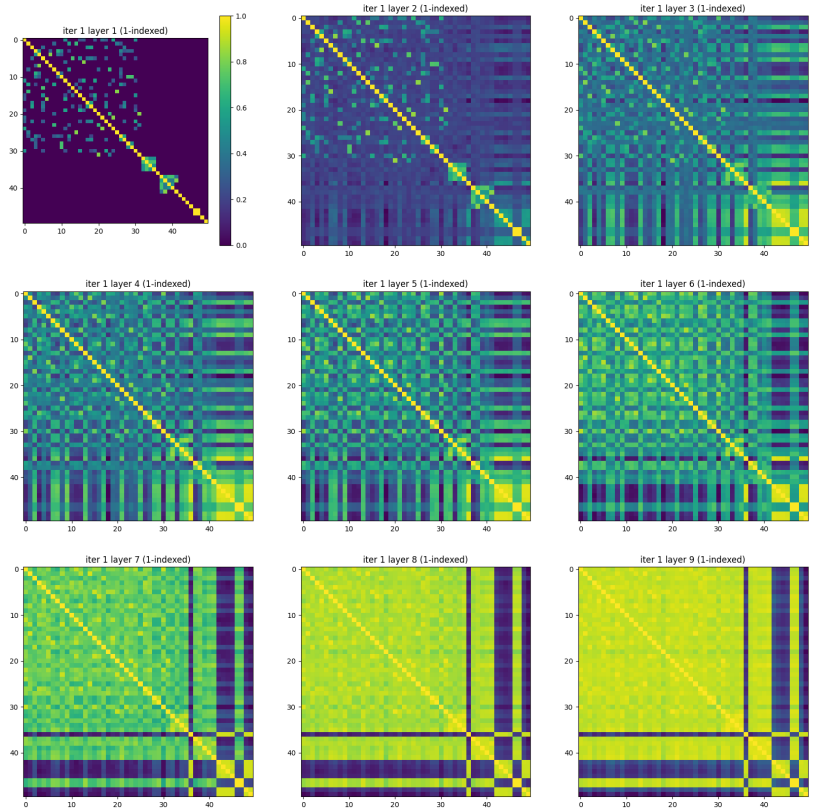
- † Dimension: $(n_{\text{nodes}} + 2) \times n_{\text{nodes}}$
- † Queries unused until prediction

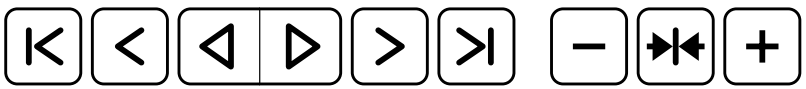
(∗) Two readout schemes:

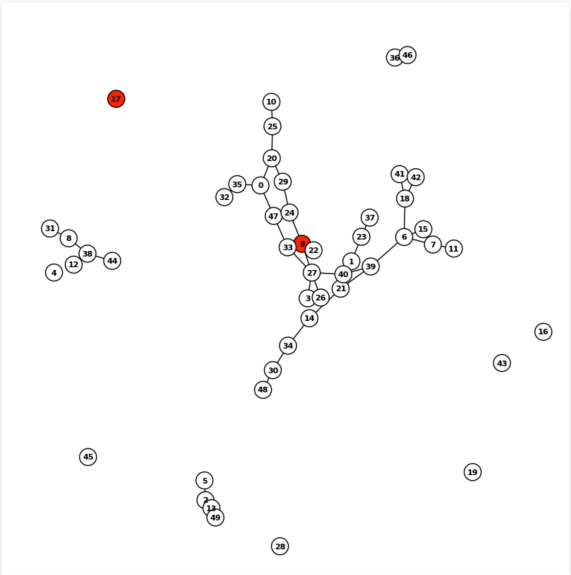
- † `inner_product`: ‘yes’ $\Leftrightarrow \langle \texttt{embd\_1}, \texttt{embd\_2} \rangle > 0$.
- † `outer_product`: one *long* MLP:
  `[embd_1 | embd_2 | (embd_1 ⊗ embd_2).flatten()]`

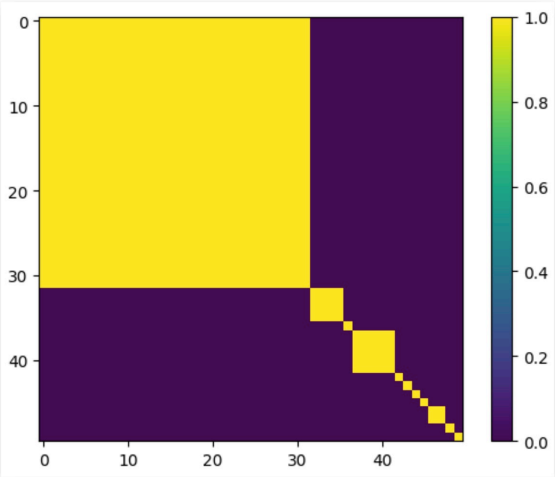
(∗) Typical training stuff... `AdamW`, weight decay, etc.

Visualizing Affinity Matrix

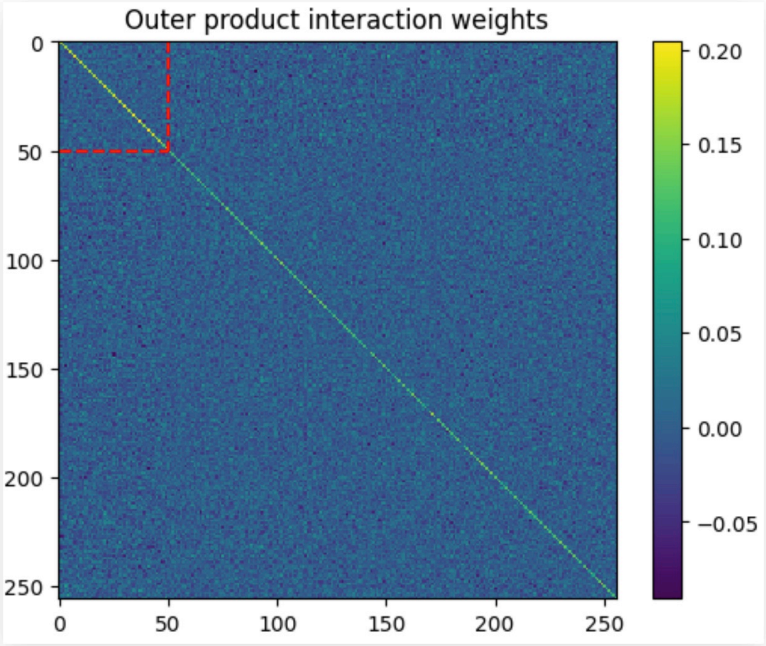


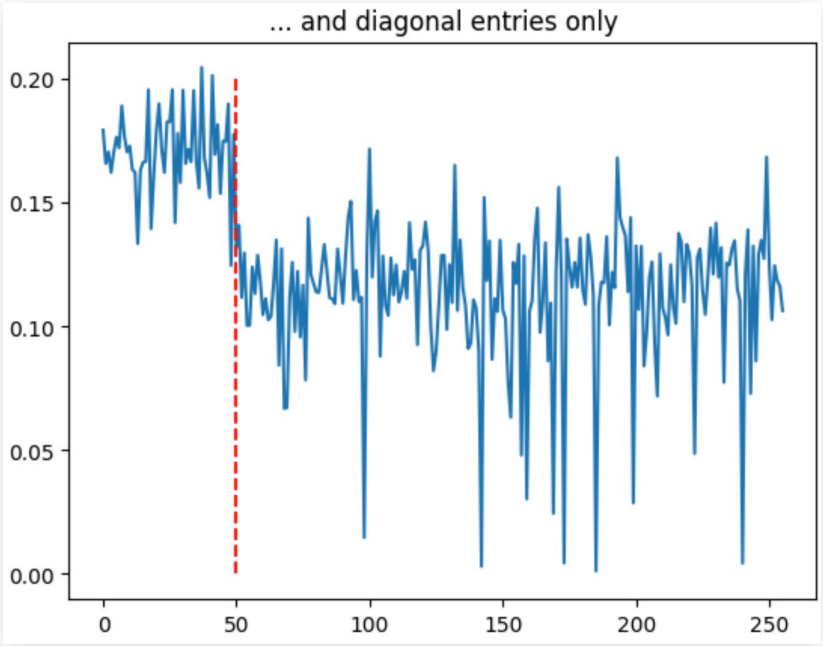




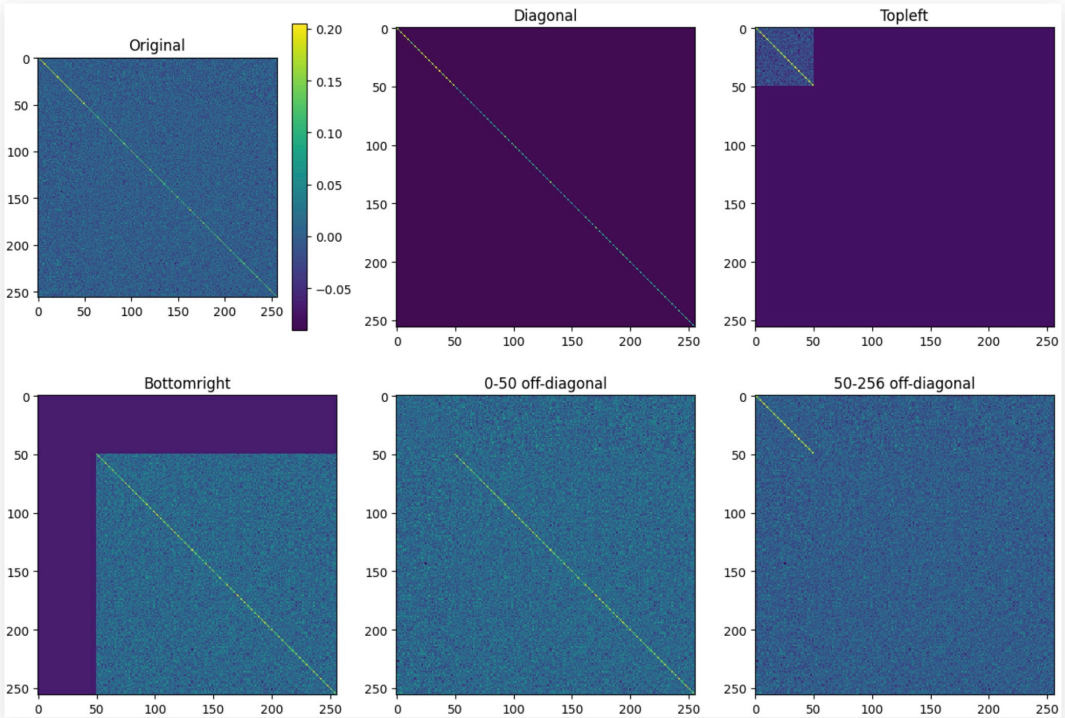


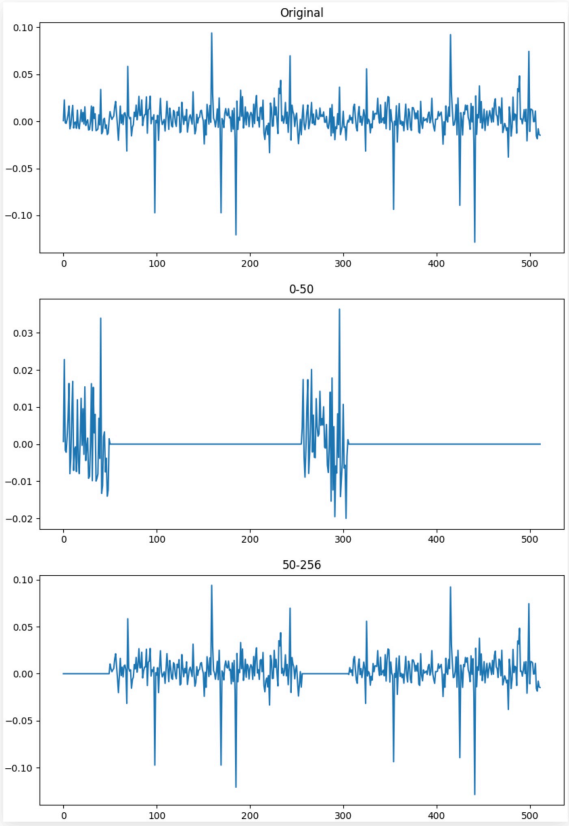
Visualizing Outer Product Interaction Weights





Visualizing Outer Product Interaction Weights: Masks





Did It Learn Too Much???

	full embd	0-50 only	50-256 only	none
full weight mat	0.992	0.991	0.989	0.990
diag only	0.923	0.473	0.920	0.338
top left	0.988	0.743	0.987	0.711
bottom right	0.994	0.992	0.993	0.992
0-50 diag off	0.993	0.990	0.990	0.990
50-256 diag off	0.991	0.994	0.992	0.991
all diag off	0.993	0.993	0.992	0.991
none	0.996	0.922	0.993	1.000

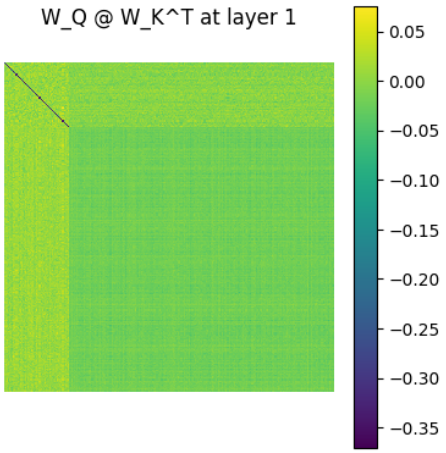
Table: $\mathbb{P}(\text{correct prediction} \mid \text{nodes are disconnected})$

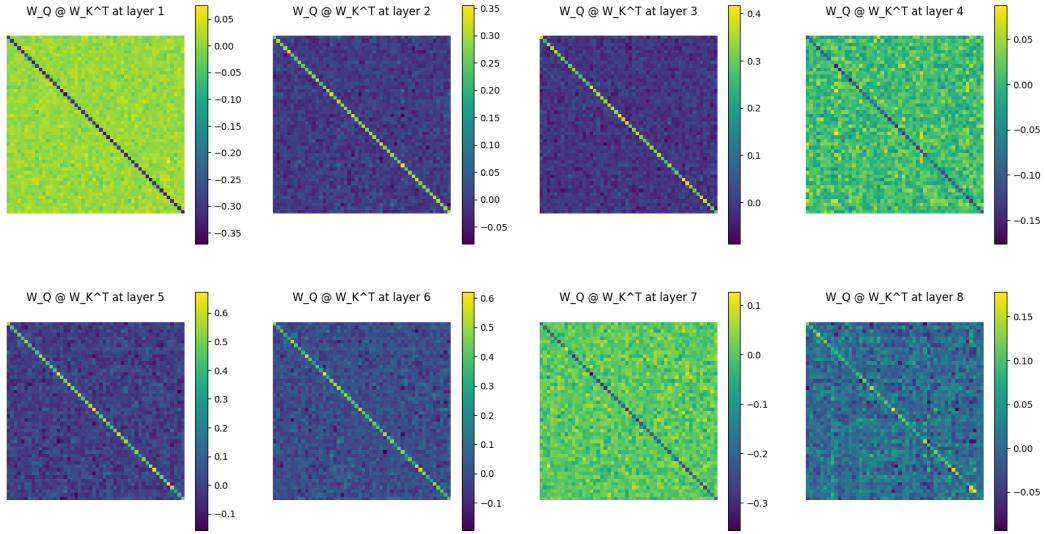
Did It Learn Too Much???

	full embd	0-50 only	50-256 only	none
full weight mat	0.996	0.998	0.998	0.997
diag only	0.998	≈ 1	0.999	1.000
top left	0.987	0.998	0.992	0.999
bottom right	0.987	0.990	0.987	0.991
0-50 diag off	0.997	0.997	0.996	0.997
50-256 diag off	0.997	0.996	0.996	0.997
all diag off	0.993	0.997	0.996	0.998
none	0.920	0.931	0.934	0

Table: $\mathbb{P}$(correct prediction | nodes are path-connected)

Looking into Self Attention

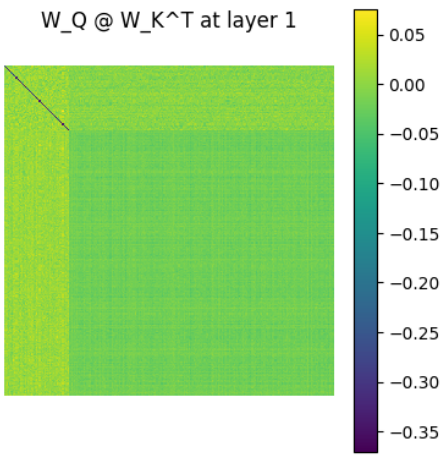


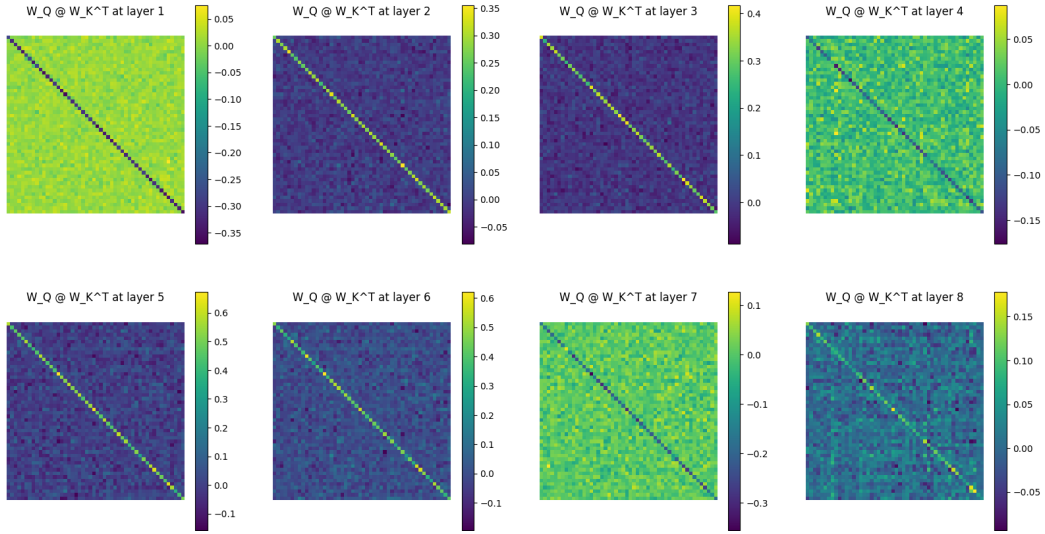


Looking into Self Attention

.

$$\tilde{\boldsymbol{h}}_i = [\mathrm{Attn}(\boldsymbol{H})]_i = \boldsymbol{h}_i + \sum_{m=1}^{n_{\text{heads}}} \sum_{j=1}^{n} \sigma(\langle \boldsymbol{Q}_m \boldsymbol{h}_i, \boldsymbol{K}_m \boldsymbol{h}_j \rangle) \cdot \boldsymbol{V}_m \boldsymbol{h}_i$$



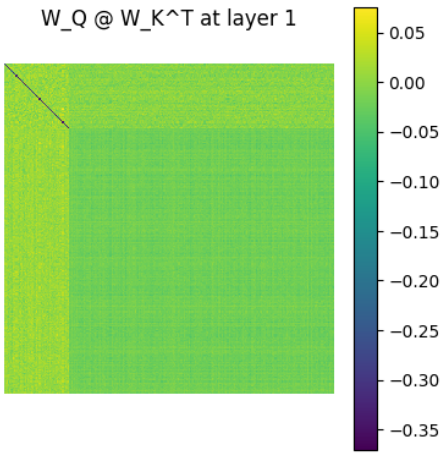


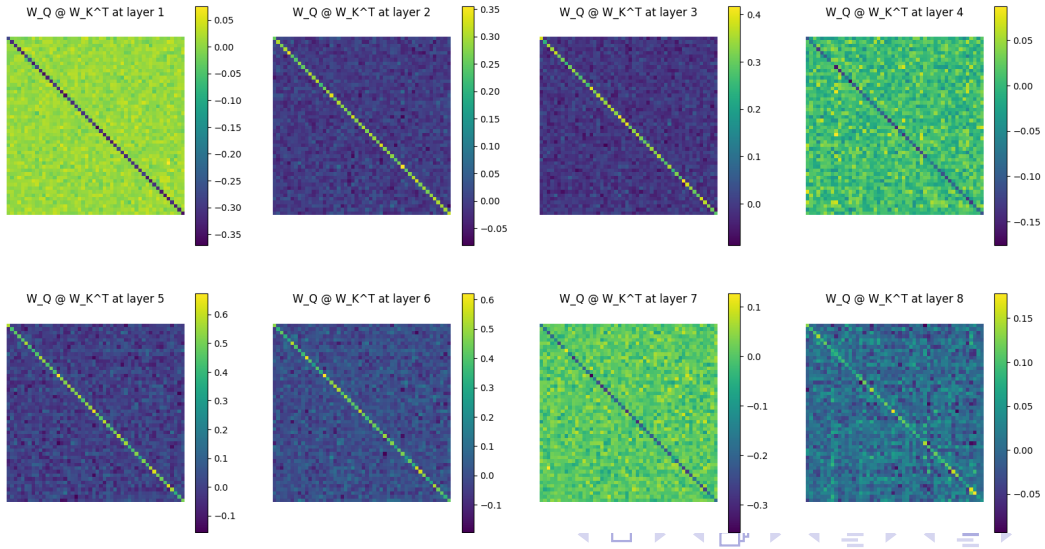
Looking into Self Attention

.

$$\tilde{\boldsymbol{h}}_i = [\mathrm{Attn}(\boldsymbol{H})]_i = \boldsymbol{h}_i + \sum_{m=1}^{n_{\text{heads}}} \sum_{j=1}^{n} \sigma(\langle \boldsymbol{Q}_m \boldsymbol{h}_i, \boldsymbol{K}_m \boldsymbol{h}_j \rangle) \cdot \boldsymbol{V}_m \boldsymbol{h}_i$$

$$\overset{n_{\text{heads}}=1}{\approx} \boldsymbol{h}_i + \sum_{j} \sigma(\langle \boldsymbol{h}_i, \boldsymbol{h}_j \rangle) \cdot \boldsymbol{V} \boldsymbol{h}_i$$





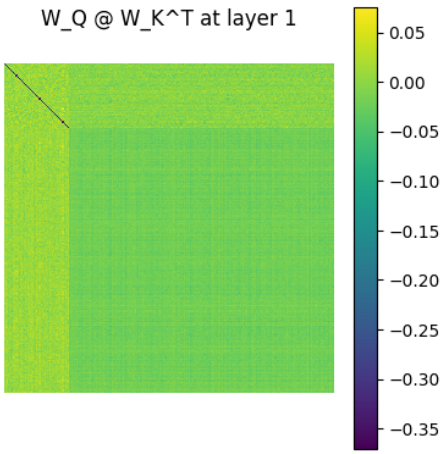
Looking into Self Attention

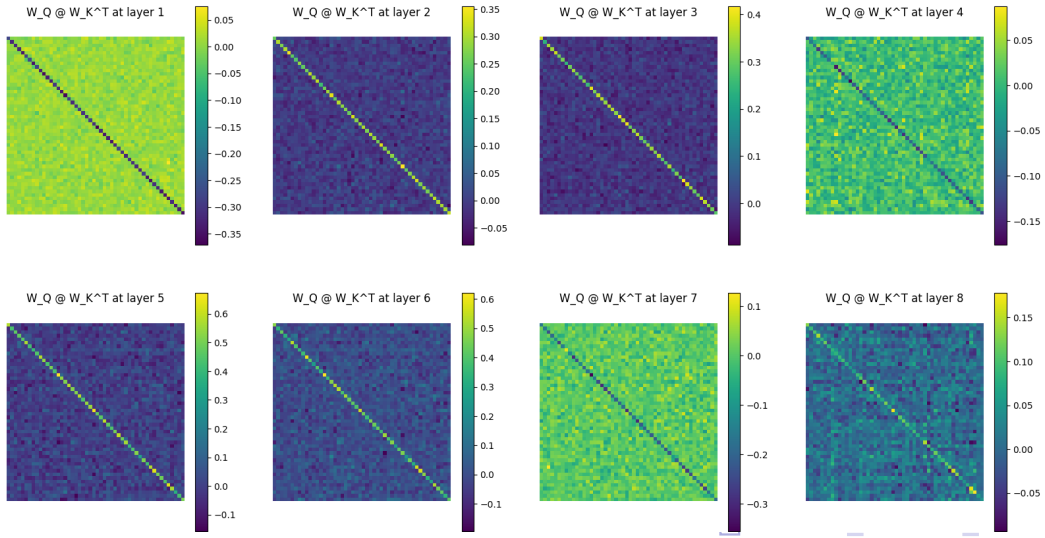
.

$$\tilde{\boldsymbol{h}}_i = [\mathrm{Attn}(\boldsymbol{H})]_i = \boldsymbol{h}_i + \sum_{m=1}^{n_{\text{heads}}} \sum_{j=1}^{n} \sigma(\langle \boldsymbol{Q}_m \boldsymbol{h}_i, \boldsymbol{K}_m \boldsymbol{h}_j \rangle) \cdot \boldsymbol{V}_m \boldsymbol{h}_i$$

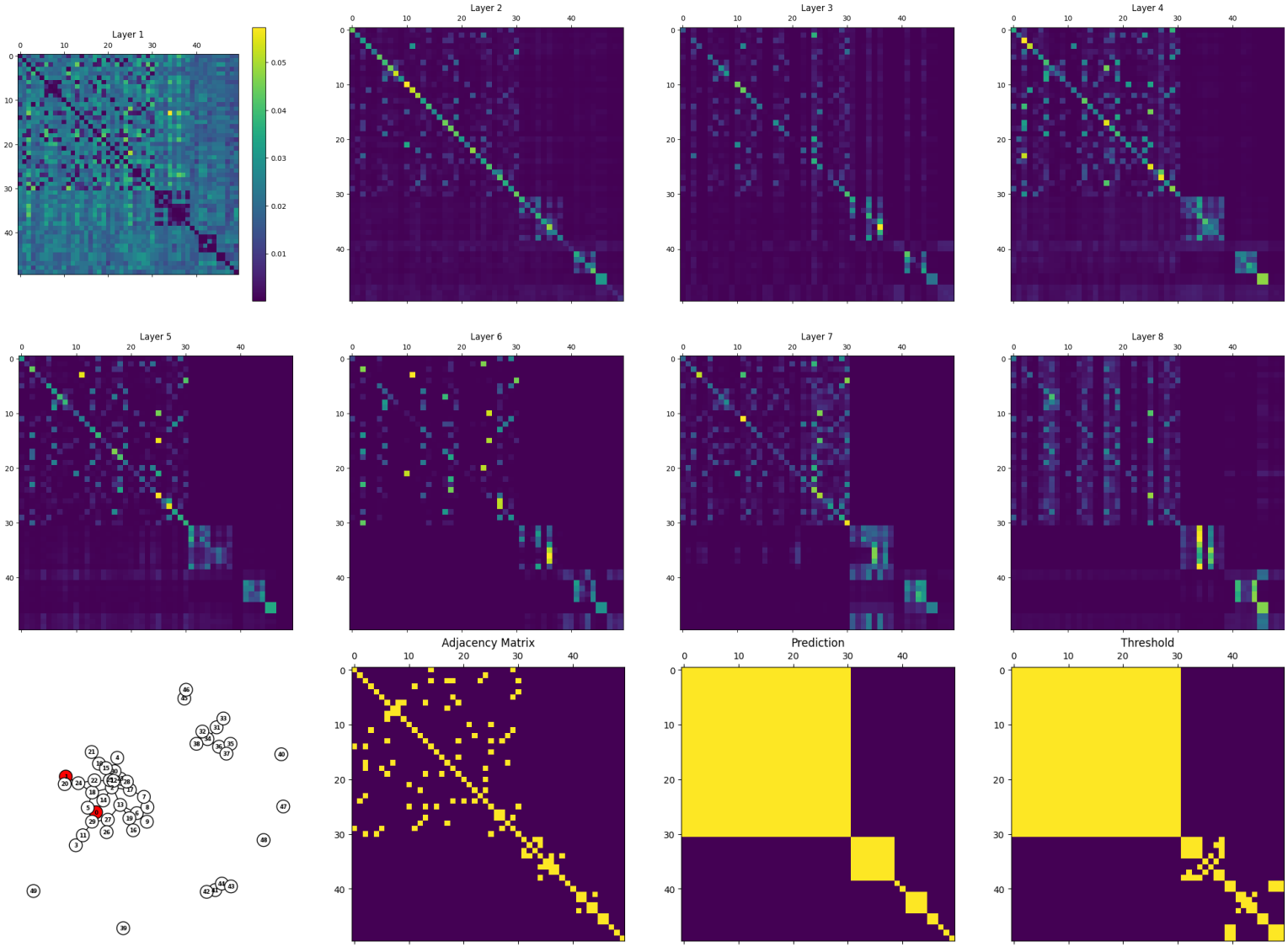
$$\overset{n_{\text{heads}}=1}{\approx} \boldsymbol{h}_i + \sum_j \sigma(\langle \boldsymbol{h}_i, \boldsymbol{h}_j \rangle) \cdot \boldsymbol{V}\boldsymbol{h}_i$$

$$\approx \boldsymbol{h}_i + \sum_{\substack{j:(i,j)\text{belong to} \\ \text{same component}}} \sigma(\langle \boldsymbol{h}_i, \boldsymbol{h}_j \rangle) \cdot \boldsymbol{V}\boldsymbol{h}_i \qquad \ldots \text{ a } \textbf{message-passing algorithm}?$$

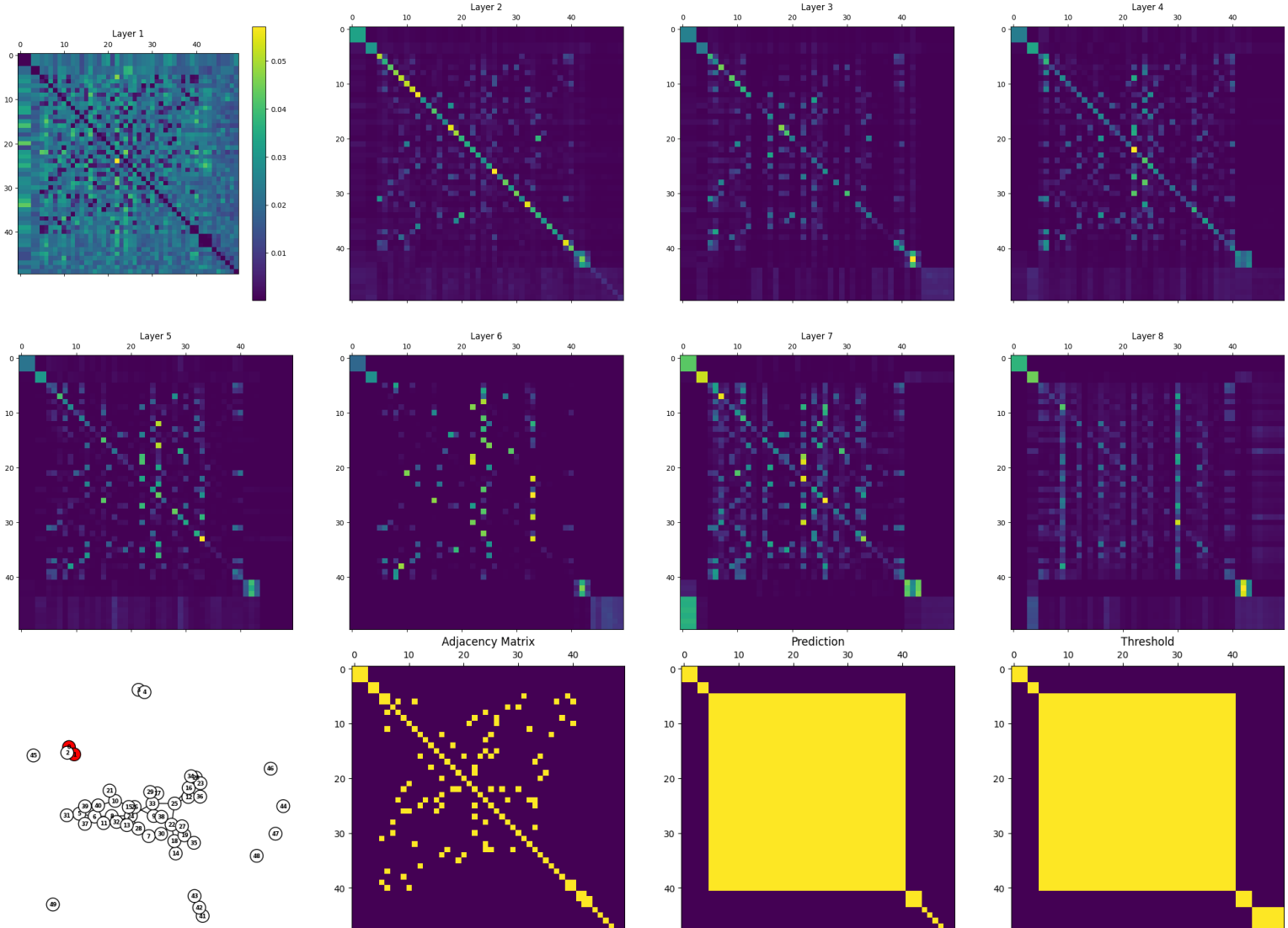




Attention Resembles Adj Matrix... Inverted???



Attention Resembles Adj Matrix... Inverted???



Next Steps

- (∗) Look into some of the raised questions
- (∗) `inner_product`, `inner_product`, any better training objectives?
- (∗) Alternative graph-generation algorithms – what is the most *canonical* way to do this?
- (∗) From qualitative to quantitative hypotheses
- (∗) Different objectives; downstream tasks

Next Steps

- (∗) Look into some of the raised questions
- (∗) `inner_product`, `inner_product`, any better training objectives?
- (∗) Alternative graph-generation algorithms – what is the most *canonical* way to do this?
- (∗) From qualitative to quantitative hypotheses
- (∗) Different objectives; downstream tasks

Next Steps

- (∗) Look into some of the raised questions
- (∗) `inner_product`, `inner_product`, any better training objectives?
- (∗) Alternative graph-generation algorithms – what is the most *canonical* way to do this?
- (∗) From qualitative to quantitative hypotheses
- (∗) Different objectives; downstream tasks

Next Steps

- (∗) Look into some of the raised questions
- (∗) `inner_product`, `inner_product`, any better training objectives?
- (∗) Alternative graph-generation algorithms – what is the most *canonical* way to do this?
- (∗) From qualitative to quantitative hypotheses
- (∗) Different objectives; downstream tasks

Next Steps

(∗) Look into some of the raised questions

(∗) `inner_product`, `inner_product`, any better training objectives?

(∗) Alternative graph-generation algorithms – what is the most *canonical* way to do this?

(∗) From qualitative to quantitative hypotheses

(∗) Different objectives; downstream tasks