
INTERPRETING AI SOLUTIONS TO THE WORD PROBLEM IN AUTOMATIC GROUPS?

Category theory, combinatorics, and machine learning workshop
Fall 2025, ICERM, *Categorification and computation in algebraic combinatorics*
September 18, 2025

J. Elisenda Grigsby
Boston College

REFERENCES (EXPOSITORY & CODE):

- Anthropic Interpretability Team, Transformer-Circuits Thread. *Mathematical Framework for Transformer Circuits (2021)*, *Toy Models of Superposition (2022)*, *Towards Monosemanticity (2023)*, *Circuit Tracing (2025)*
 - Bill Casselman, *Computation in Coxeter Groups I: Multiplication*. Electronic Journal of Combinatorics, Vol. 9, 2002
 - Neel Nanda, *TransformerLens*, GitHub Repository, <https://github.com/TransformerLensOrg>;
 - Neel Nanda, *A Comprehensive Mechanistic Interpretability Explainer & Glossary*, <https://www.neelnanda.io/mechanistic-interpretability/glossary>
 - Chris Olah. *Understanding LSTM Networks* (blog post), 2015
 - Neel Nanda, Lawrence Chan, Tom Lieberum, Jess Smith, Jacob Steinhardt. *Progress Measures for Grokking via Mechanistic Interpretability*. ICLR 2023
 - Bilal Chughtai, Lawrence Chan, Neel Nanda. *A Toy Model of Universality: Reverse Engineering how Networks Learn Group Operations*. ICML 2023
 - Dashiell Stander, Qinan Yu, Honglu Fan, Stella Biderman. *Grokking Group Multiplication with Cosets*. ICML 2024
-

WHAT THIS TALK IS ABOUT:

WHAT THIS TALK IS ABOUT:

- Publicizing mechanistic interpretability as a (young) empirical and theoretical field

WHAT THIS TALK IS ABOUT:

- Publicizing mechanistic interpretability as a (young) empirical and theoretical field
 - Interesting math problems there
 - That field might be interested in yours
-

WHAT THIS TALK IS ABOUT:

- Publicizing mechanistic interpretability as a (young) empirical and theoretical field
 - Interesting math problems there
 - That field might be interested in yours
 - Publicizing mentorship/team-building as a way to do this work
-

MOTIVATING QUESTION:

MOTIVATING QUESTION:

CAN MACHINE LEARNING MODELS BUILD
(GEOMETRIC) INTUITION?

IF SO, HOW?

GROUPS AS LANGUAGES:

GROUPS AS LANGUAGES:

Finitely-presented group:

$$G = \langle s_1, \dots, s_n \mid r_1, \dots, r_m \rangle$$

GROUPS AS LANGUAGES:

Finitely-presented group:

$$G = \langle s_1, \dots, s_n \mid r_1, \dots, r_m \rangle$$

Given w and w' two *words* in generators and their inverses, do they represent the same element of the group?

GROUPS AS LANGUAGES:

Finitely-presented group:

$$G = \langle s_1, \dots, s_n \mid r_1, \dots, r_m \rangle$$

Given w and w' two *words* in generators and their inverses, do they represent the same element of the group?

Equivalently, does $w^{-1}w' = e$, the identity element of the group?

GROUPS AS LANGUAGES:

Finitely-presented group:

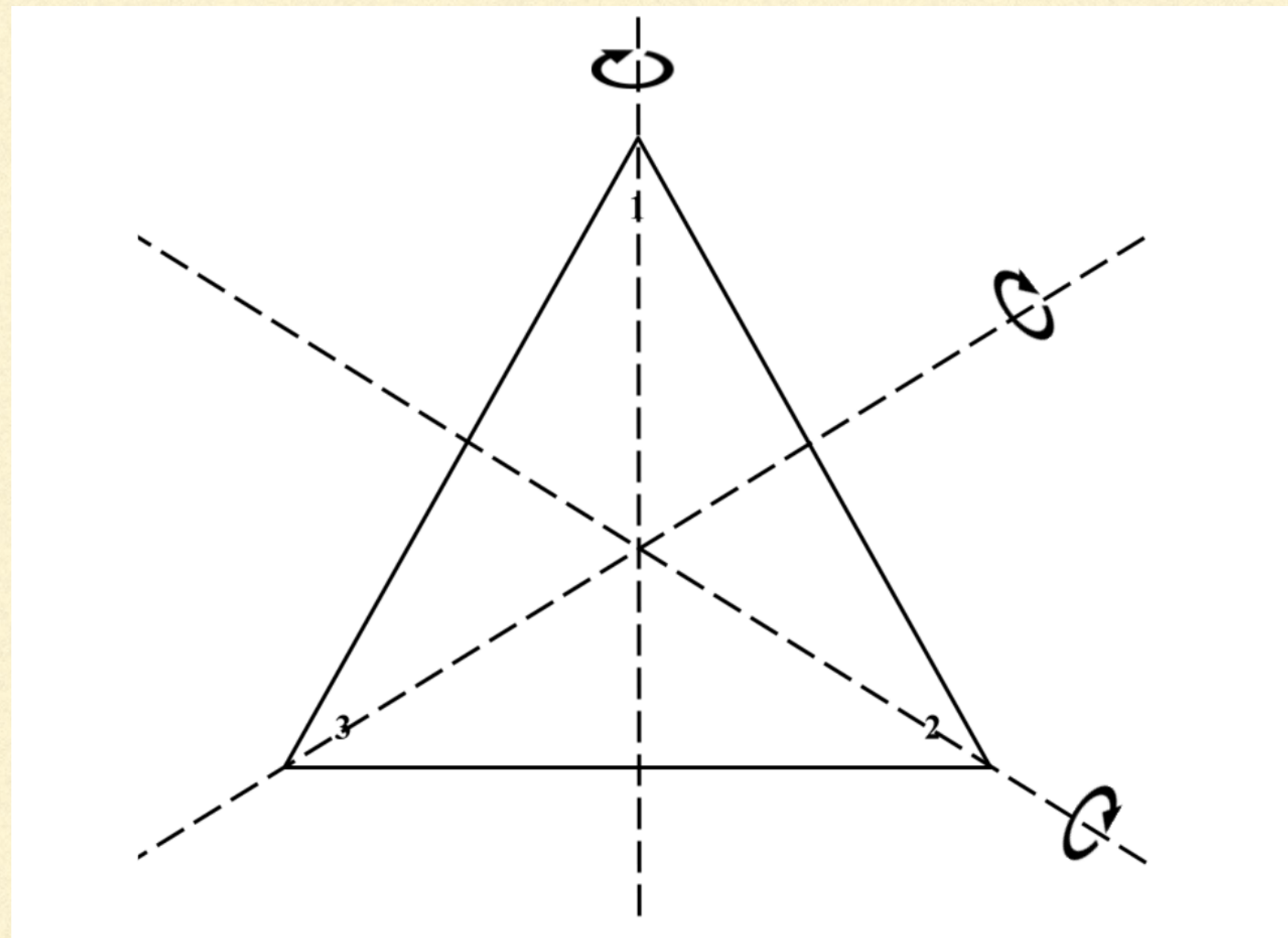
$$G = \langle s_1, \dots, s_n \mid r_1, \dots, r_m \rangle$$

Given w and w' two words in generators and their inverses, do they represent the same element of the group?

Equivalently, does $w^{-1}w' = e$, the identity element of the group?

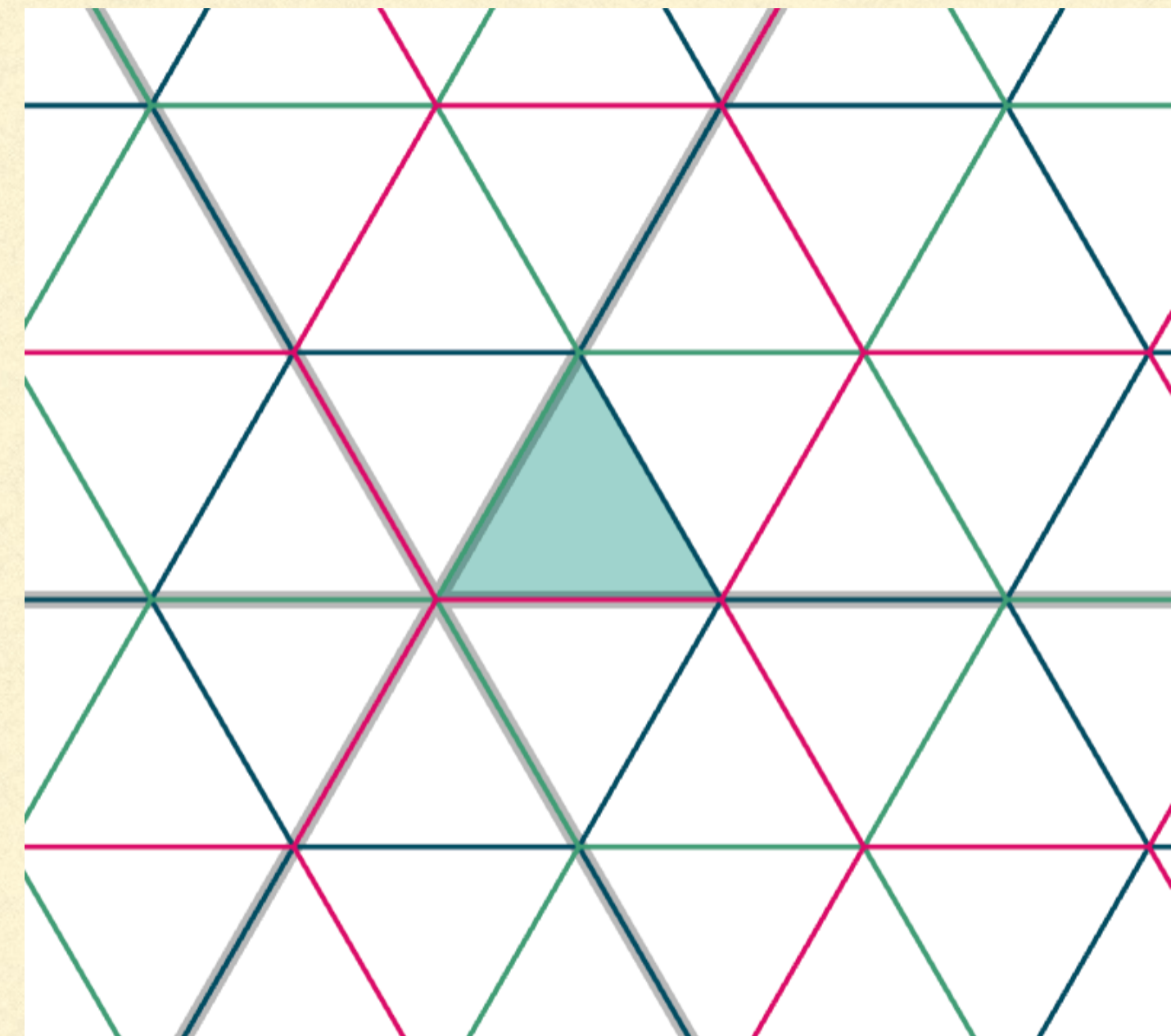
Hard (sometimes impossible) combinatorial problem in general

GROUPS (OFTEN) COME FROM GEOMETRY:



S_3

Image credit: BCExpMMLL team



$\tilde{A}_2$

Image credit: Joel Gibson, *LieVis*

OUTLINE:

1. Supervised learning using neural network models:
Sequence processing
 2. Mechanistic interpretability: key structural ideas and
empirical methods
 3. Case study: the word problem in Coxeter groups
-

OUTLINE:

1. Supervised learning using neural network models:
Sequence processing
 2. Mechanistic interpretability: key structural ideas and empirical methods
 3. Case study: the word problem in Coxeter groups
-

Supervised learning

- 1) A finite *labeled data set*
- $$\mathcal{D} = \left\{ \left(\underbrace{x^{(i)}}_{\substack{\text{inputs} \\ \downarrow \\ \mathbb{R}^{n_{in}}}}, \underbrace{y^{(i)}}_{\substack{\text{outputs} \\ \downarrow \\ \mathbb{R}^{n_{out}}}} \right) \right\}_{i=1}^N$$
- 2) Parameterized function class
- $$\mathcal{F} = \{ h_{\theta} : \mathbb{R}^{n_{in}} \rightarrow \mathbb{R}^{n_{out}} \mid \theta \in (\Omega = \mathbb{R}^D) \}$$
- Parameter space
↓
-

Supervised learning

- 1) A finite *labeled data set*
- $$\mathcal{D} = \left\{ \left(\underset{\substack{\text{inputs} \\ \downarrow \\ \cap \\ \mathbb{R}^{n_{in}}}}{x^{(i)}}, \underset{\substack{\text{outputs} \\ \downarrow \\ \cap \\ \mathbb{R}^{n_{out}}}}{y^{(i)}} \right) \right\}_{i=1}^N$$
- 2) Parameterized function class
- $$\mathcal{F} = \{ h_{\theta} : \mathbb{R}^{n_{in}} \rightarrow \mathbb{R}^{n_{out}} \mid \theta \in (\Omega = \mathbb{R}^D) \}$$
- Parameter space
↓
- Task: Using $\mathcal{D}$, learn the *optimal* function**
 $h_{\theta} : \mathbb{R}^{n_{in}} \rightarrow \mathbb{R}^{n_{out}}$ allowing us to predict y from x

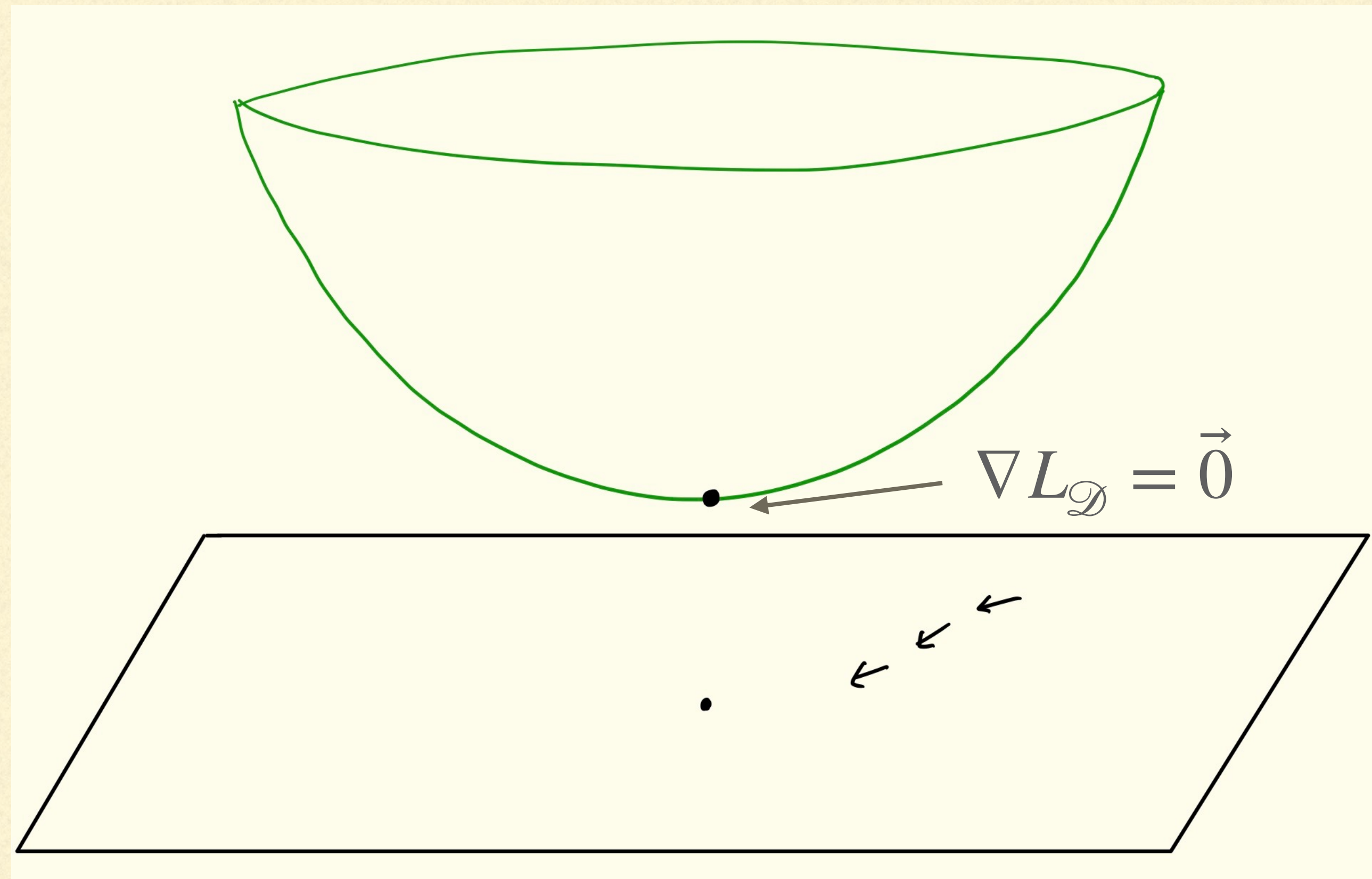
Supervised learning

- 1) A finite *labeled data set*
- $$\mathcal{D} = \left\{ \left(\underbrace{x^{(i)}}_{\substack{\text{inputs} \\ \downarrow \\ \mathbb{R}^{n_{in}}}}, \underbrace{y^{(i)}}_{\substack{\text{outputs} \\ \downarrow \\ \mathbb{R}^{n_{out}}}} \right) \right\}_{i=1}^N$$
- 2) Parameterized function class
- $$\mathcal{F} = \{ h_{\theta} : \mathbb{R}^{n_{in}} \rightarrow \mathbb{R}^{n_{out}} \mid \theta \in (\Omega = \mathbb{R}^D) \}$$
- Parameter space
↓

Gradient descent on appropriate *Error function* $L_{\mathcal{D}}(\theta)$

Geometry of the loss landscape

Gradient
Descent



Graph of
 $L_{\mathcal{D}} : \mathbb{R}^D \rightarrow \mathbb{R}$

Parameter space
 $\Omega = \mathbb{R}^D$

Supervised learning for sequential inputs

A finite *labeled data set*

$$\mathcal{D} = \left\{ \left(\overset{\text{inputs}}{\downarrow} \underset{\mathbb{R}^{n_{in} \times L}}{X^{(i)}}, \overset{\text{outputs}}{\downarrow} \underset{\mathbb{R}^{n_{out} \times L}}{Y^{(i)}} \right) \right\}_{i=1}^N$$

$$X^{(i)} = [x_1^{(i)} \quad \dots \quad x_L^{(i)}]$$

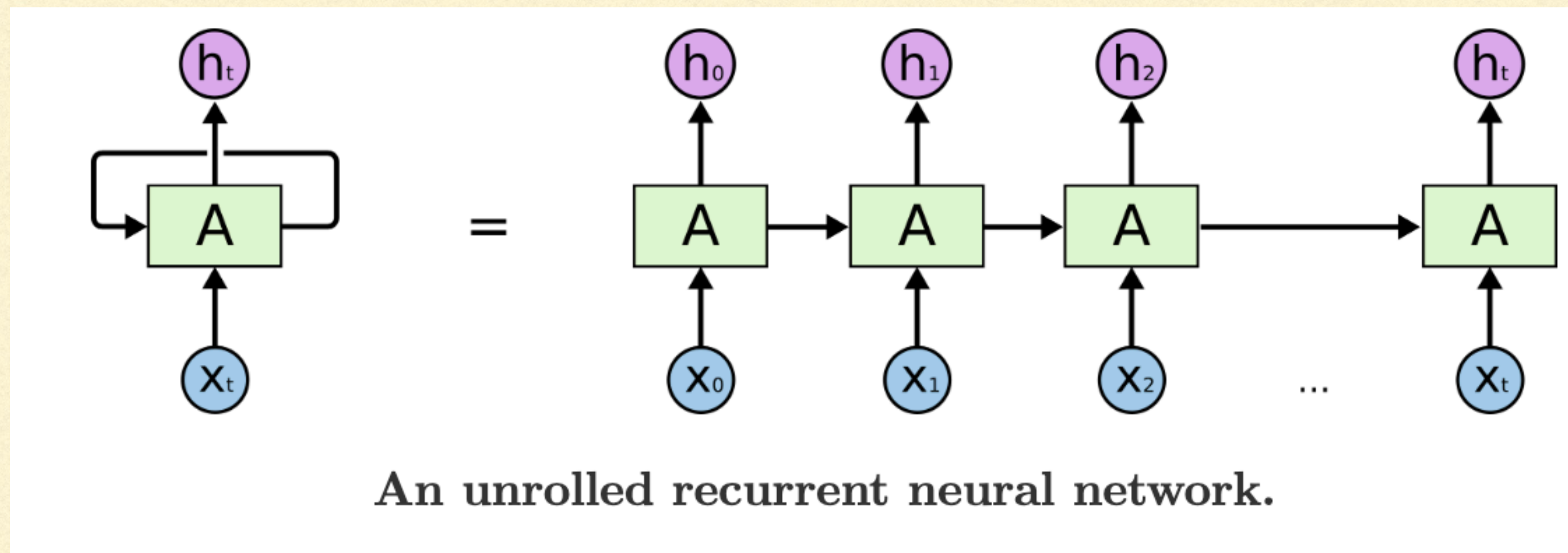
time steps
in sequence

$$Y^{(i)} = [y_1^{(i)} \quad \dots \quad y_L^{(i)}]$$

L = Context length

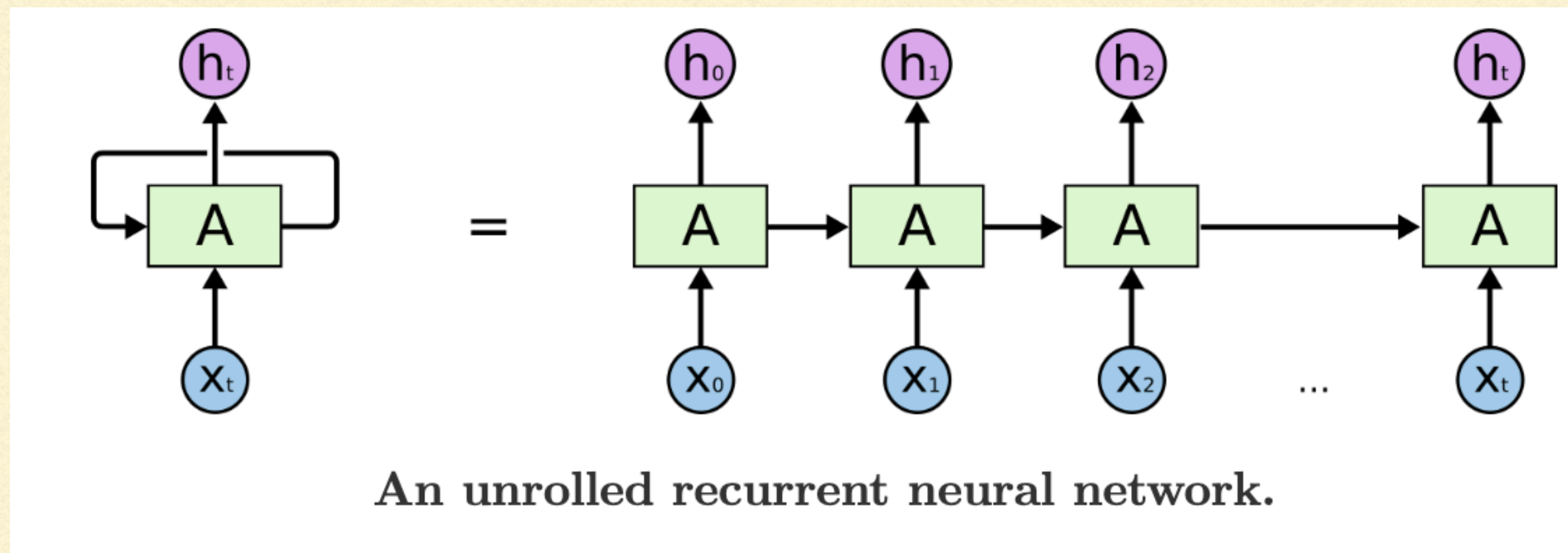
OLDER SEQUENCE-PROCESSING ARCHITECTURES:

(Vanilla) Recurrent Neural Network:



OLDER SEQUENCE-PROCESSING ARCHITECTURES:

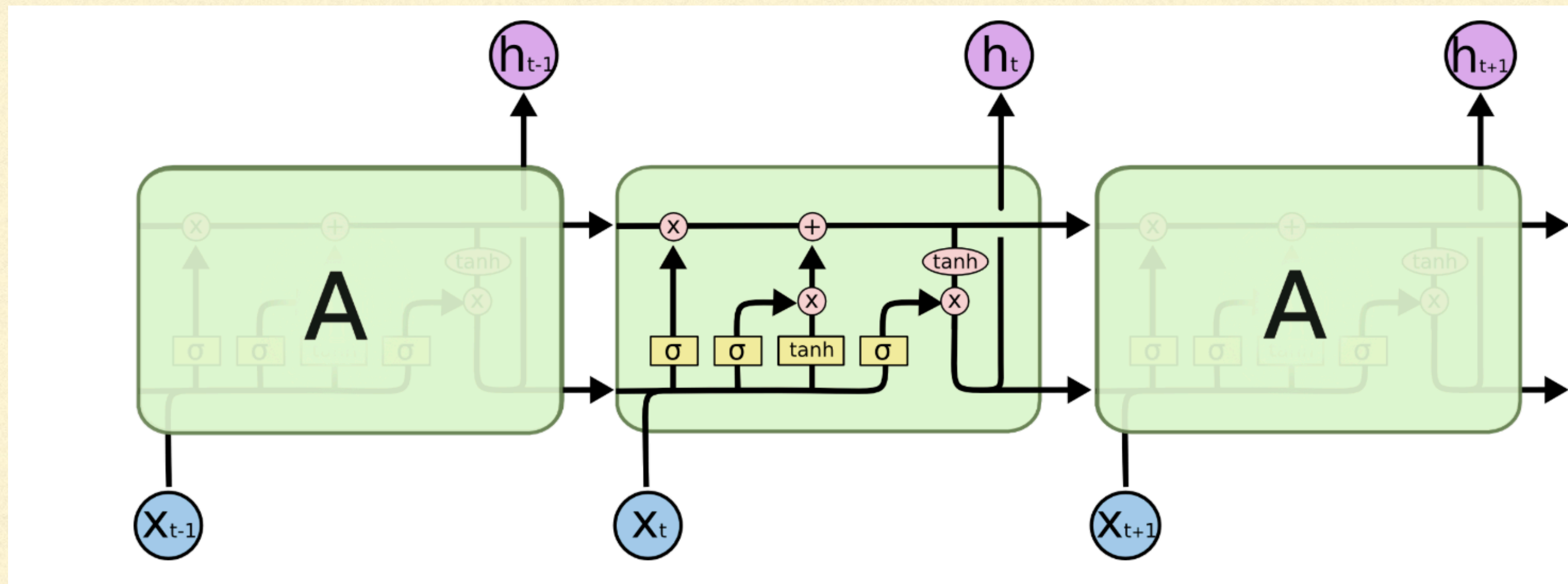
(Vanilla) Recurrent Neural Network:



The trained/learned function A computes the output h_t at position t using only an auxiliary output c_{t-1} at the previous position and the input x_t at the current position

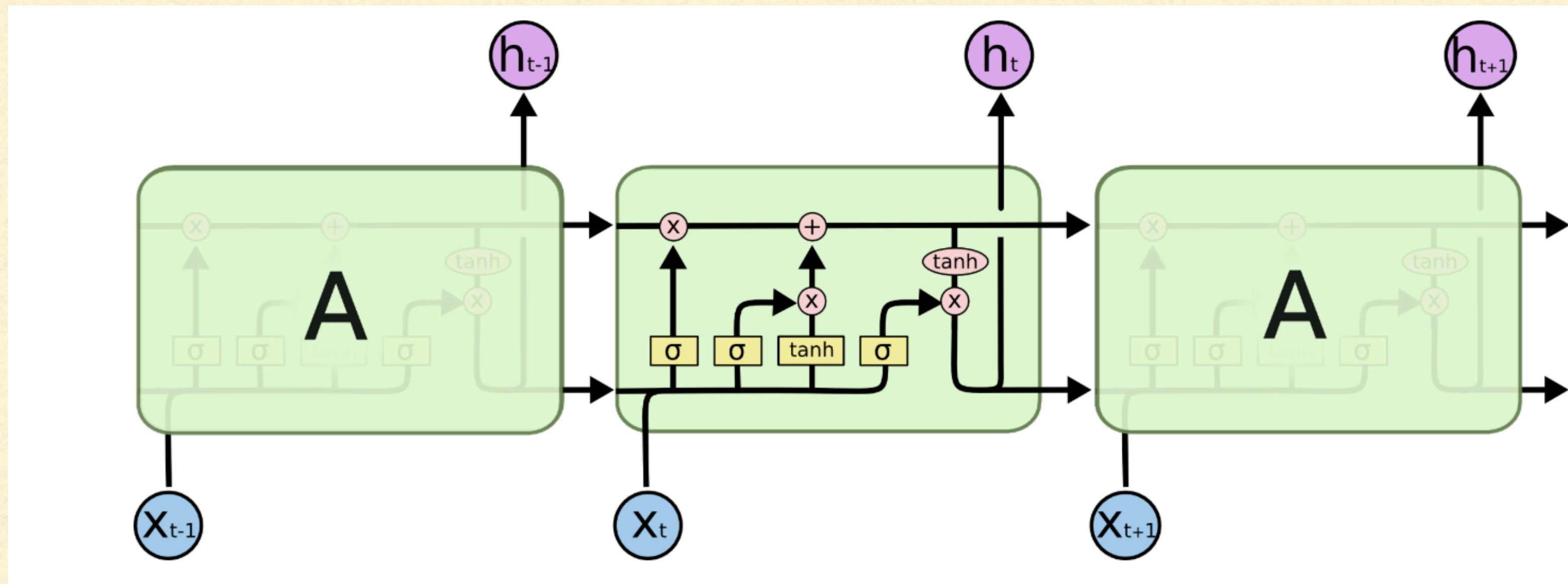
OLDER SEQUENCE-PROCESSING ARCHITECTURES:

Long Short Term Memory Network (LSTM):



OLDER SEQUENCE-PROCESSING ARCHITECTURES:

Long Short Term Memory Network (LSTM):

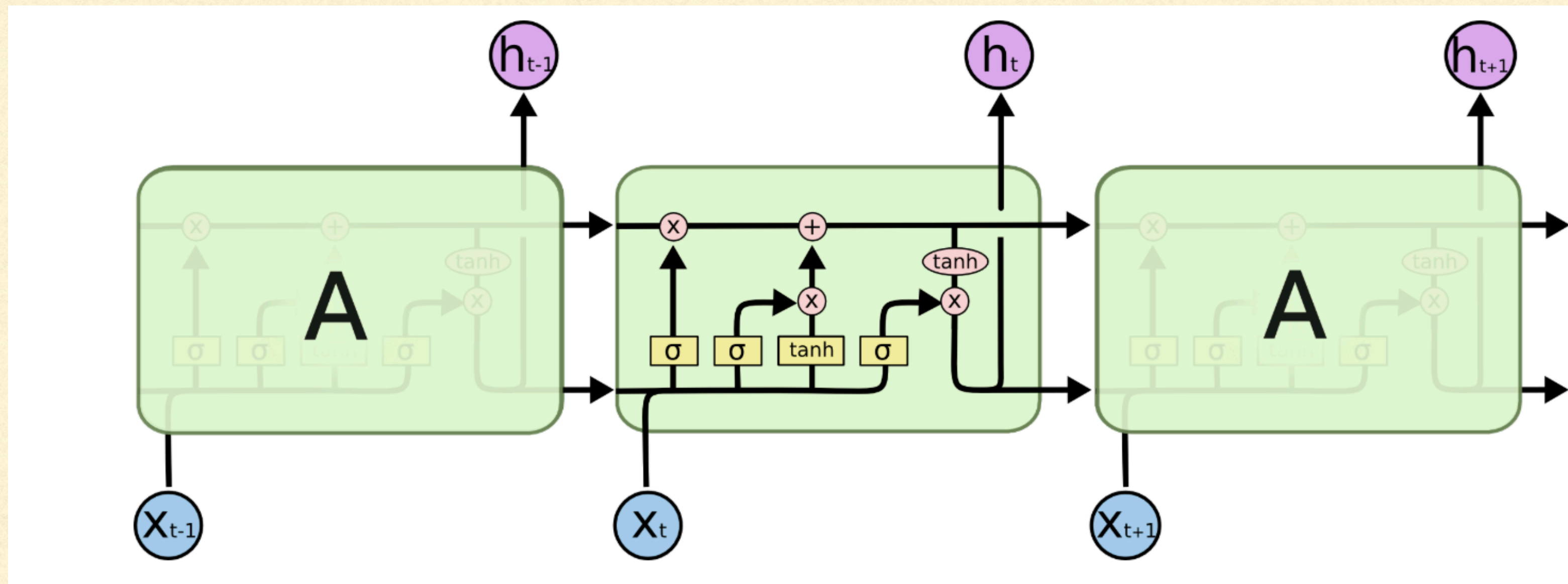


The trained/learned function A computes the output h_t at position t using:

- a cell state c_{t-1} (Long term memory)
- the output h_{t-1} at the previous position (Short term memory)
- the input x_t at the current position

OLDER SEQUENCE-PROCESSING ARCHITECTURES:

Long Short Term Memory Network (LSTM):



Part of the computation that the computational block A learns to perform is how to update its cell state (Long term memory)

$$c_{t-1} \rightarrow c_t$$

in response to new input x_t

OUTLINE:

1. Supervised learning using neural network models:
Sequence processing
 2. Mechanistic interpretability: key structural ideas and
empirical methods
 3. Case study: the word problem in Coxeter groups
-

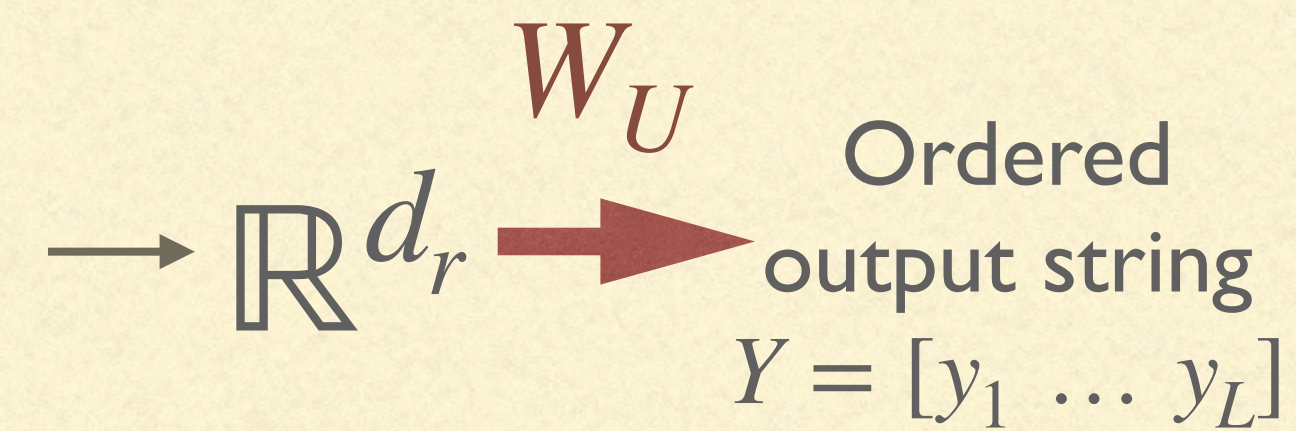
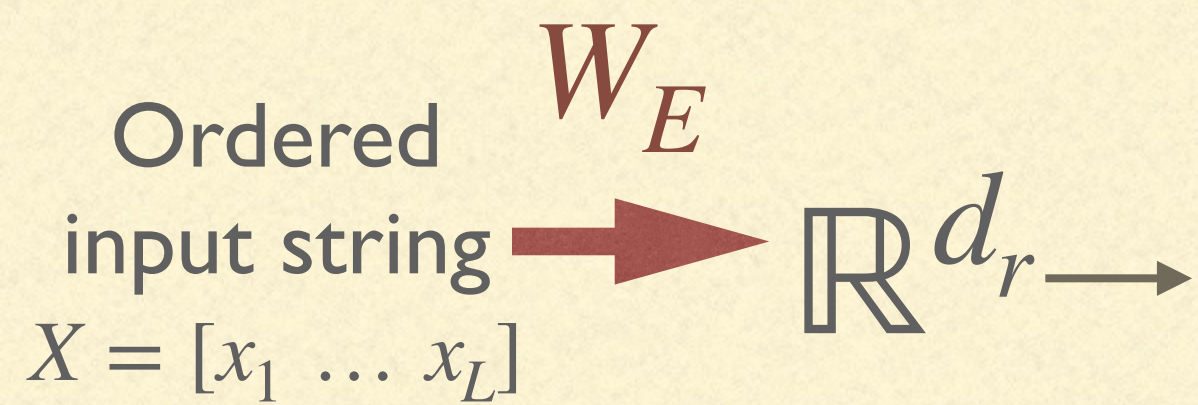
TRANSFORMERS: (E.G., GENERATIVE LLM'S)

TRANSFORMERS: (E.G., GENERATIVE LLM'S)

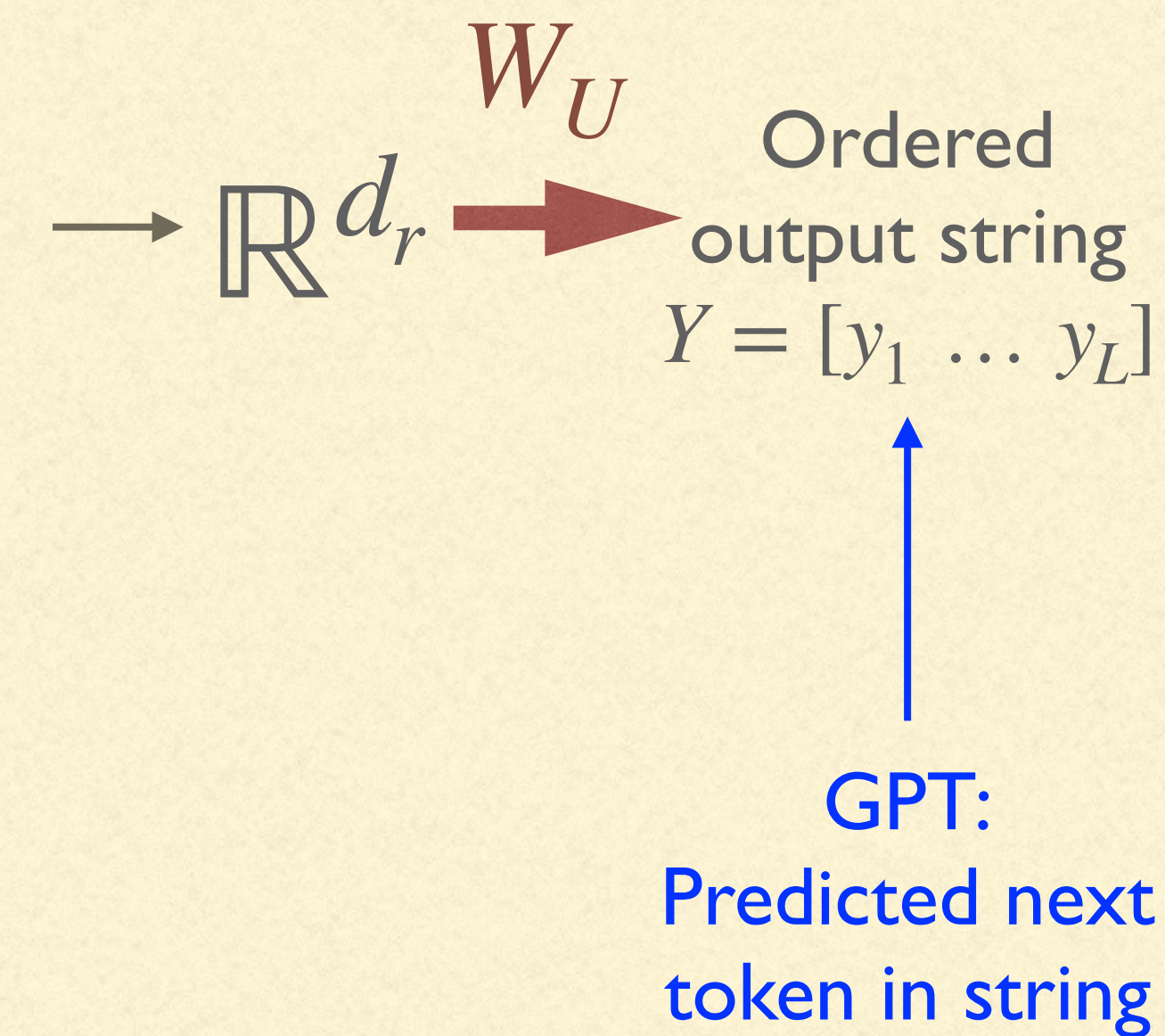
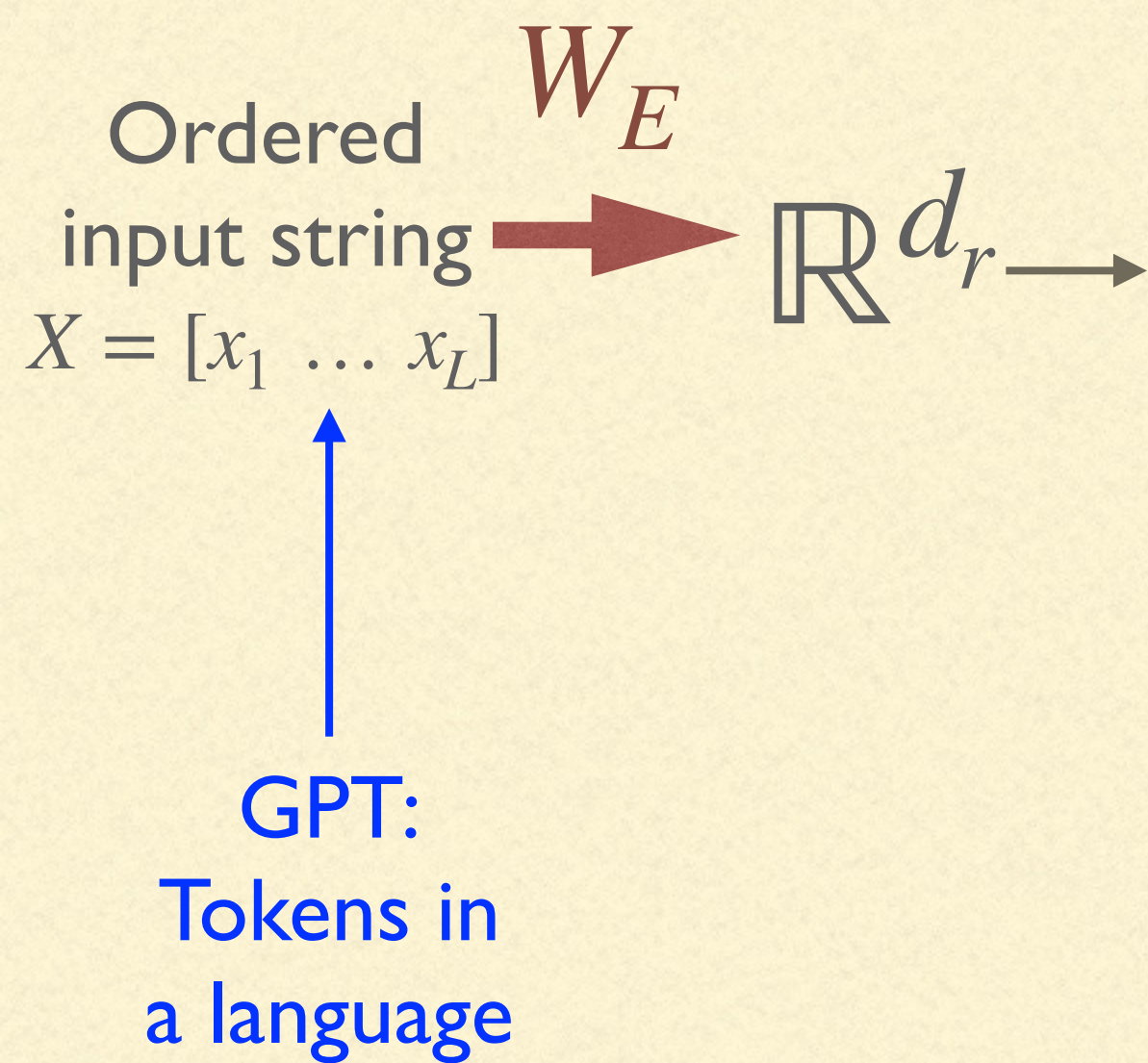
Ordered
input string
 $X = [x_1 \dots x_L]$

Ordered
output string
 $Y = [y_1 \dots y_L]$

TRANSFORMERS: (E.G., GENERATIVE LLM'S)



TRANSFORMERS: (E.G., GENERATIVE LLM'S)



TRANSFORMERS:

Ordered
input string
 $X = [x_1 \dots x_L]$

$\xrightarrow{W_E}$

$\mathbb{R}^{d_r}$

$\longrightarrow$

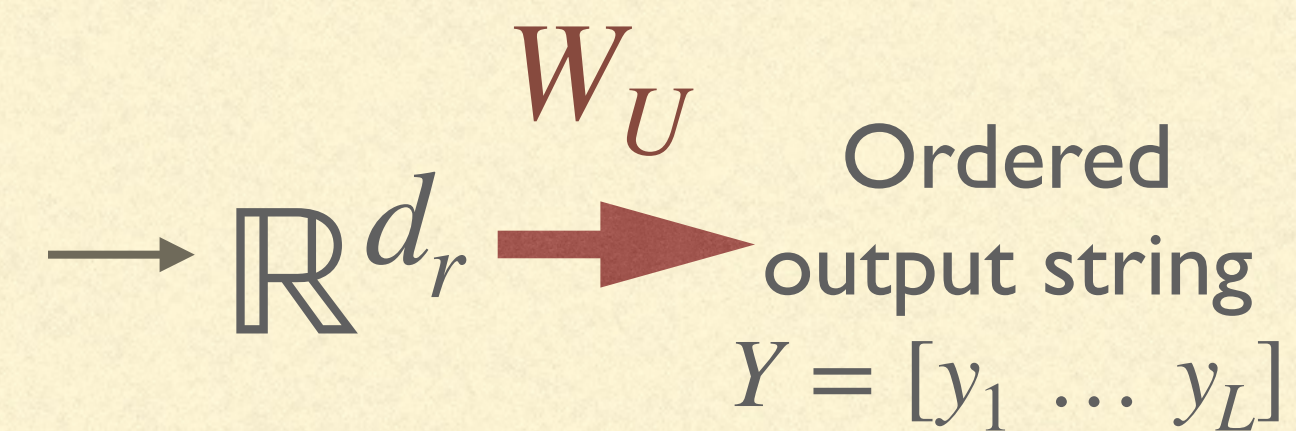
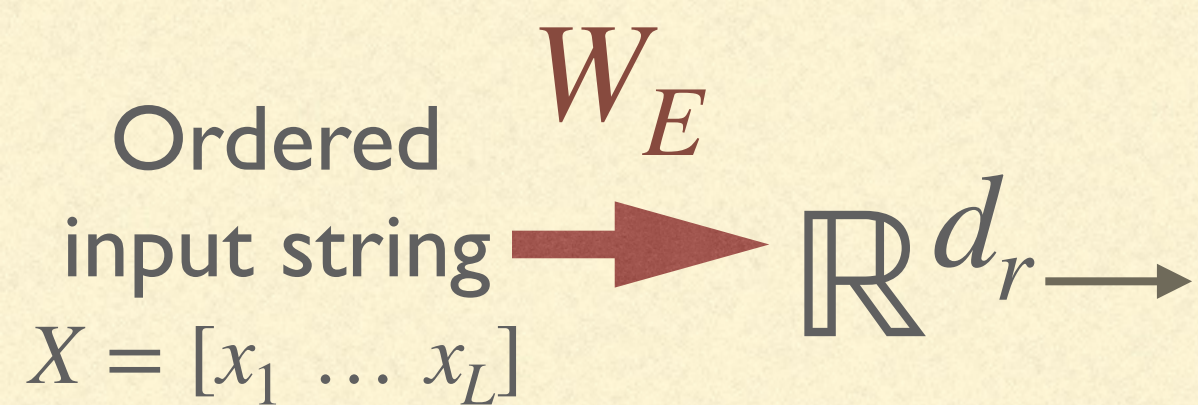
$\longrightarrow$

$\mathbb{R}^{d_r}$

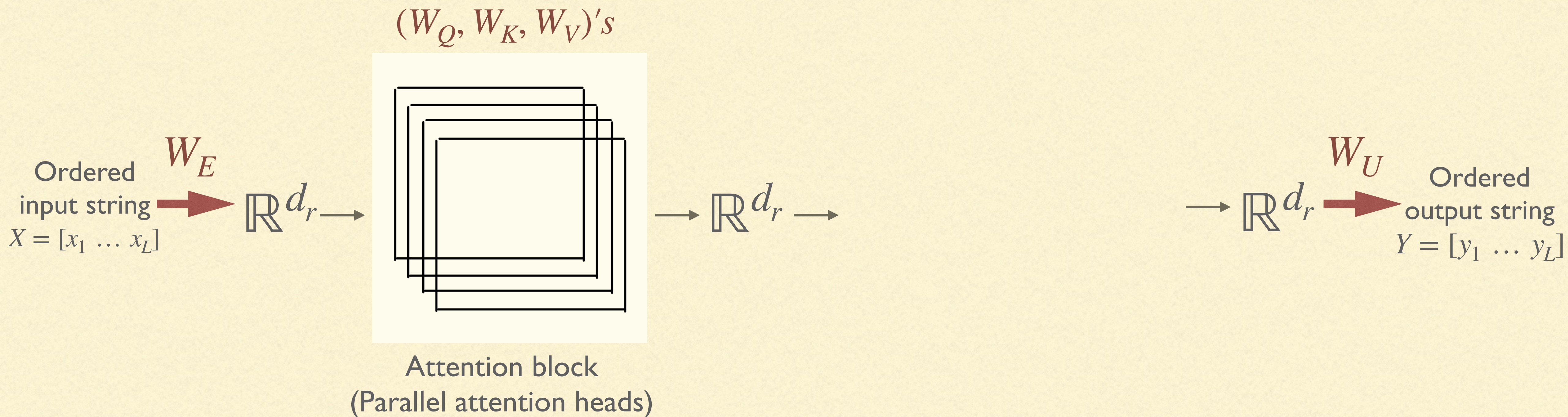
$\xrightarrow{W_U}$

Ordered
output string
 $Y = [y_1 \dots y_L]$

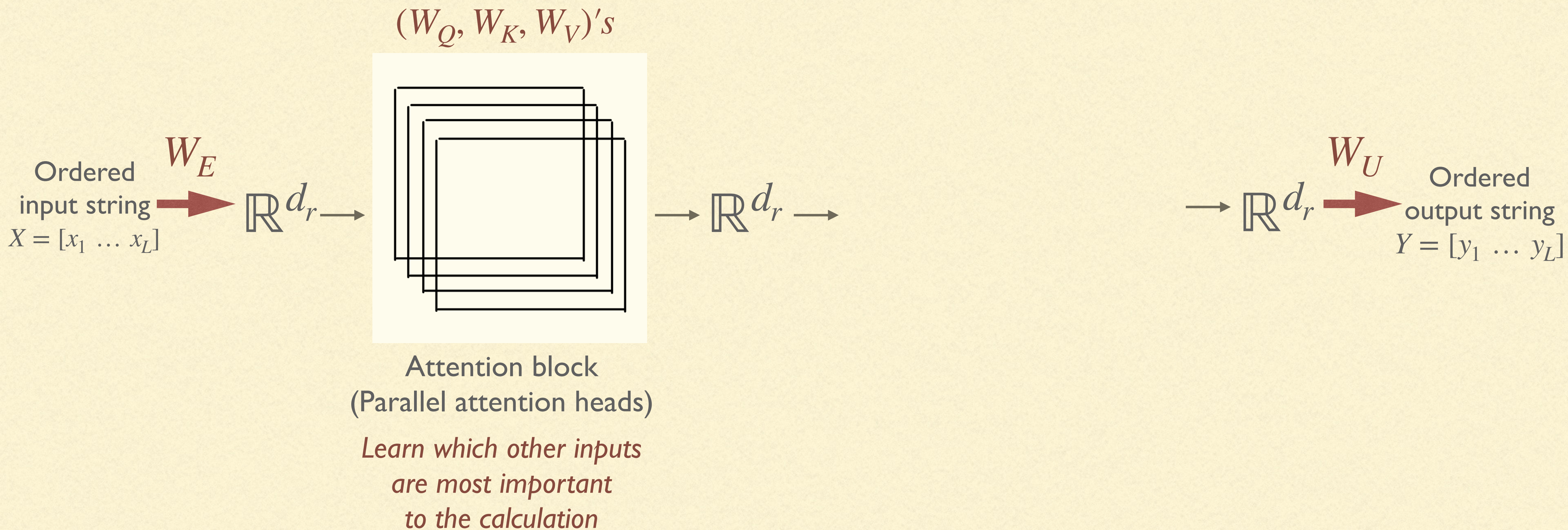
TRANSFORMERS (+ATTENTION BLOCKS):



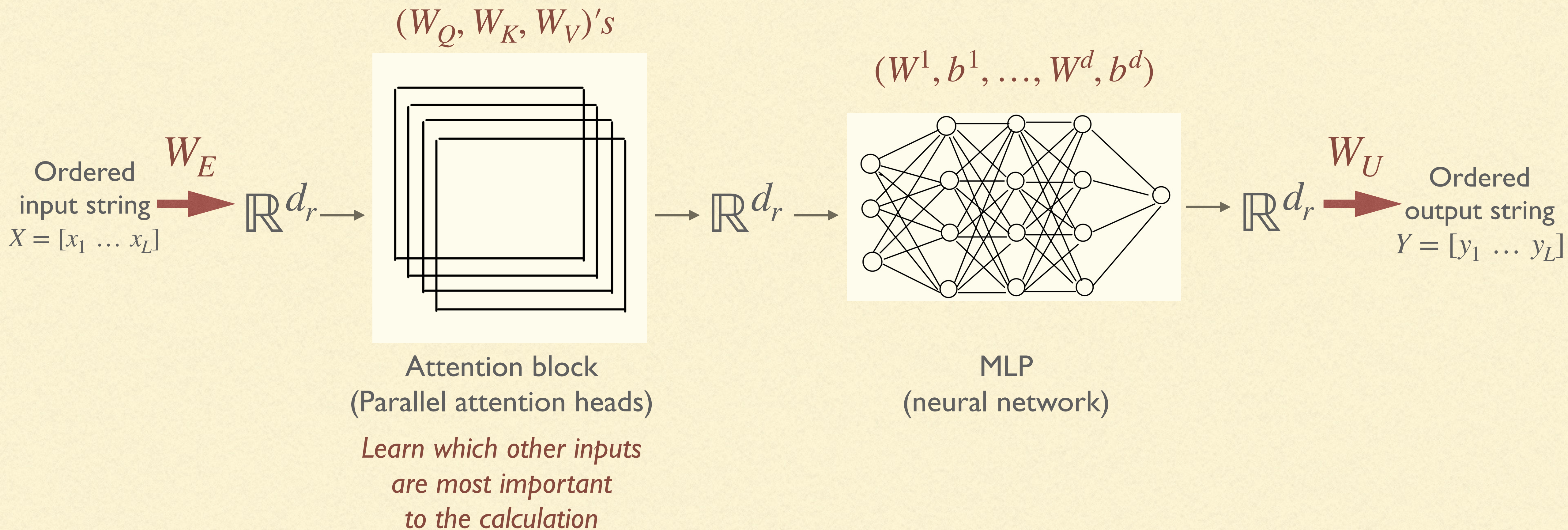
TRANSFORMERS (+ATTENTION BLOCKS):



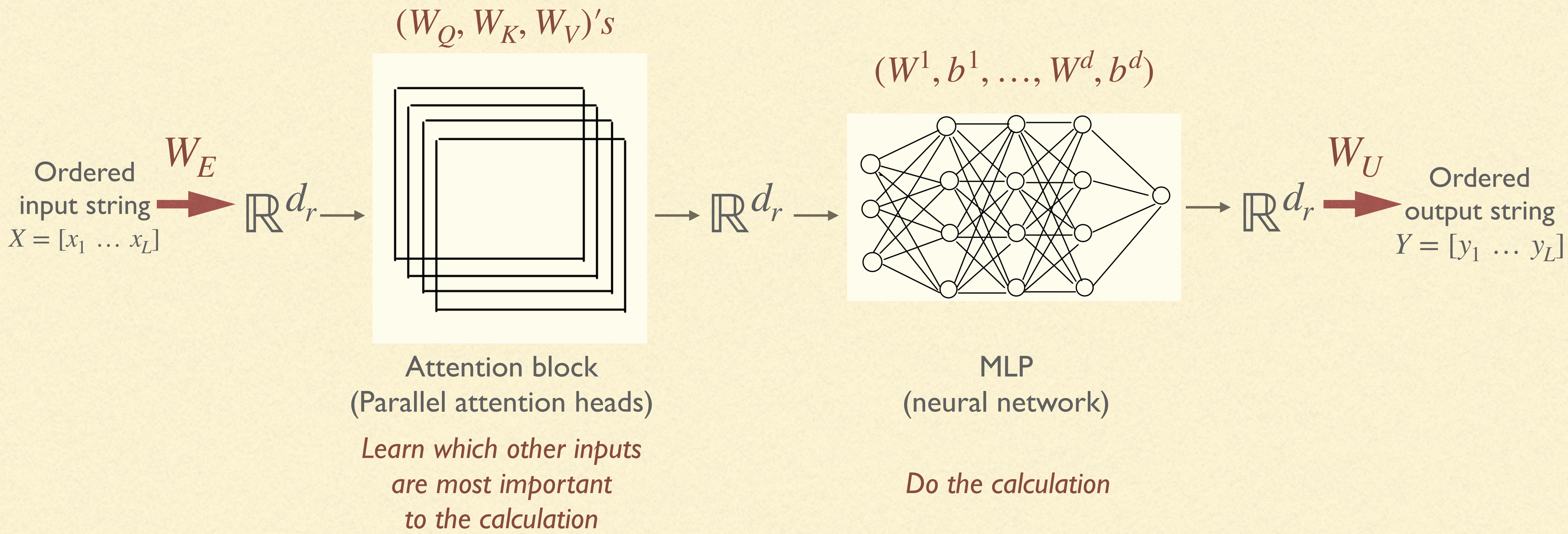
TRANSFORMERS (+ATTENTION BLOCKS):



TRANSFORMERS (+ATTENTION BLOCKS):



TRANSFORMERS (+ATTENTION BLOCKS):



ALIGNMENT AND MONOSEMANTICITY:

(Anthropic interpretability team, Chris Olah, Neel Nanda, many many others)

ALIGNMENT AND MONOSEMANTICITY:

(Anthropic interpretability team, Chris Olah, Neel Nanda, many many others)

Desirable situation for interpretability:

- *Alignment* of a model's computational units & human-interpretable concepts

ALIGNMENT AND MONOSEMANTICITY:

(Anthropic interpretability team, Chris Olah, Neel Nanda, many many others)

Desirable situation for interpretability:

- *Alignment* of a model's computational units & human-interpretable concepts

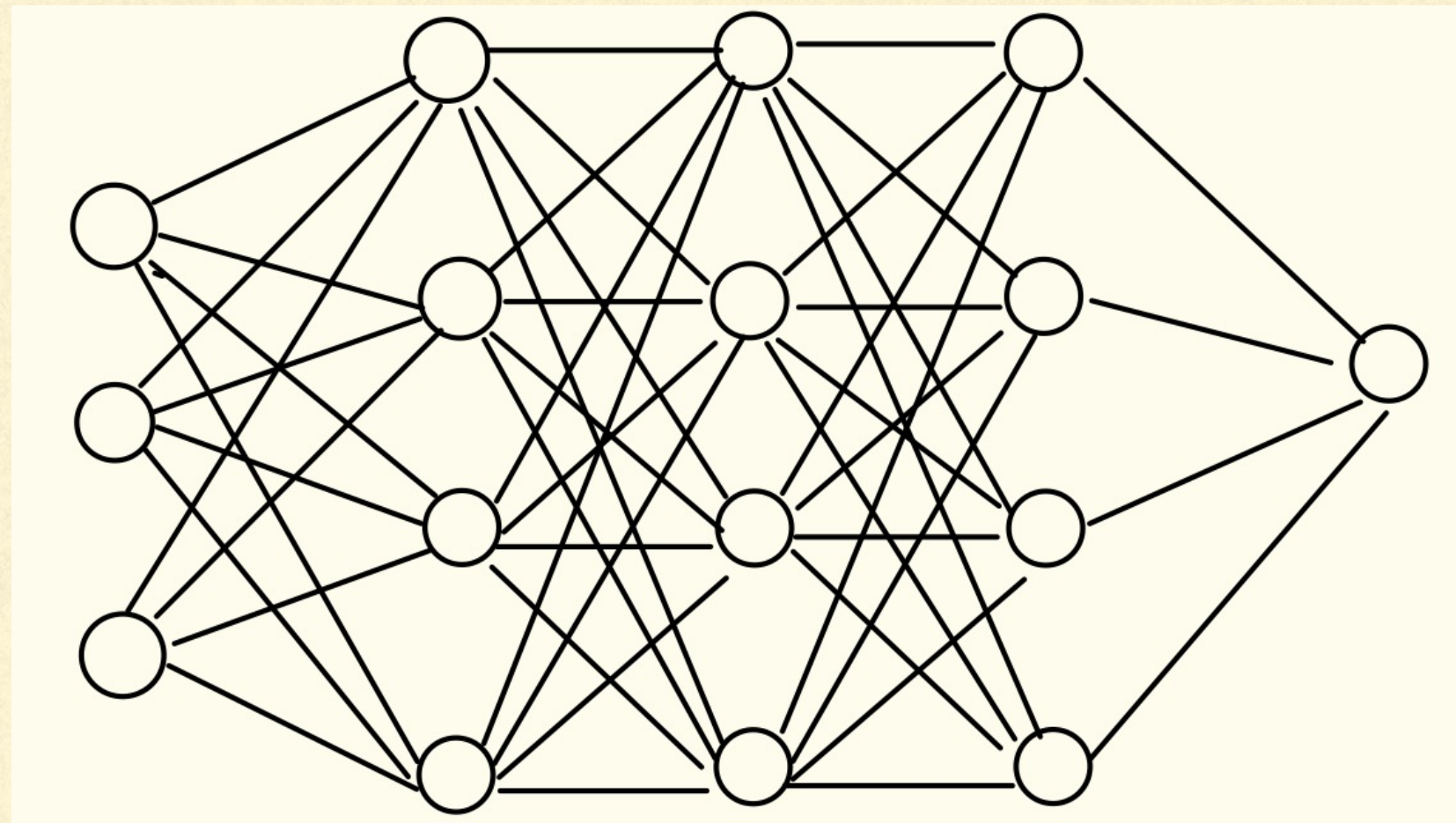


ALIGNMENT AND MONOSEMANTICITY:

(Anthropic interpretability team, Chris Olah, Neel Nanda, many many others)

Desirable situation for interpretability:

- *Alignment* of a model's computational units & human-interpretable concepts

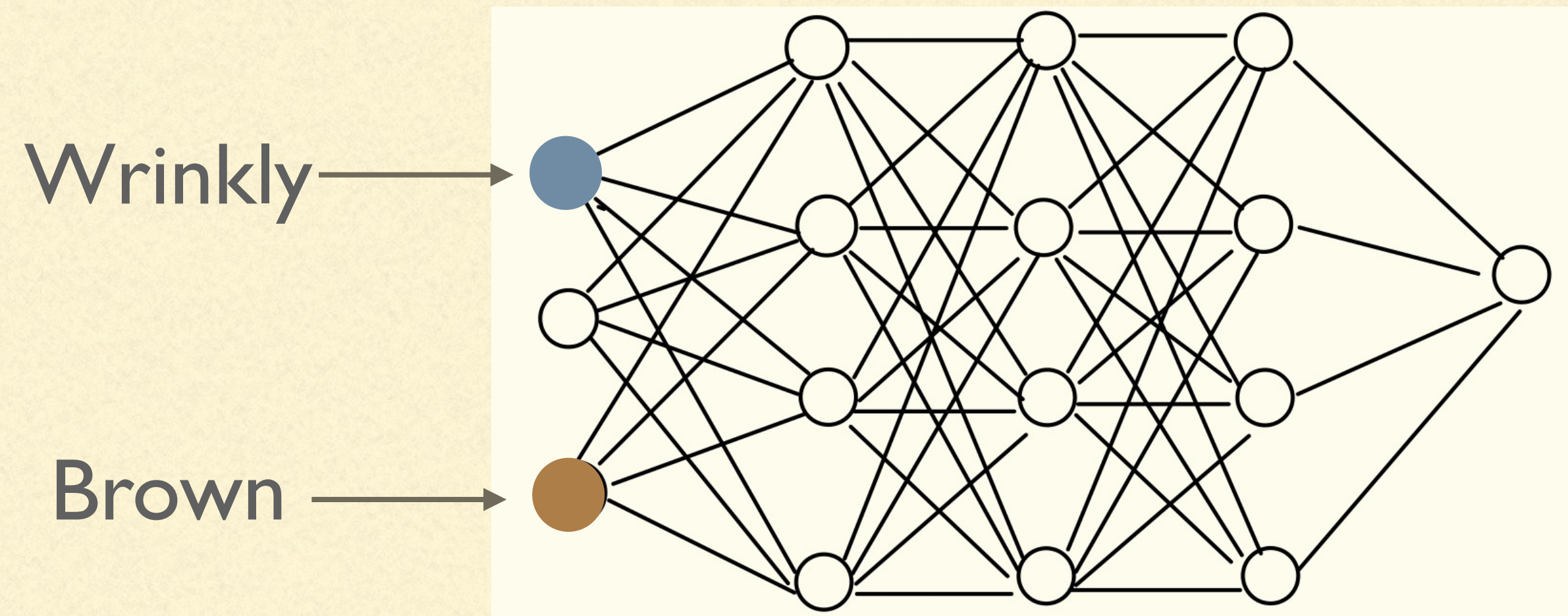


ALIGNMENT AND MONOSEMANTICITY:

(Anthropic interpretability team, Chris Olah, Neel Nanda, many many others)

Desirable situation for interpretability:

- *Alignment* of a model's computational units & human-interpretable concepts

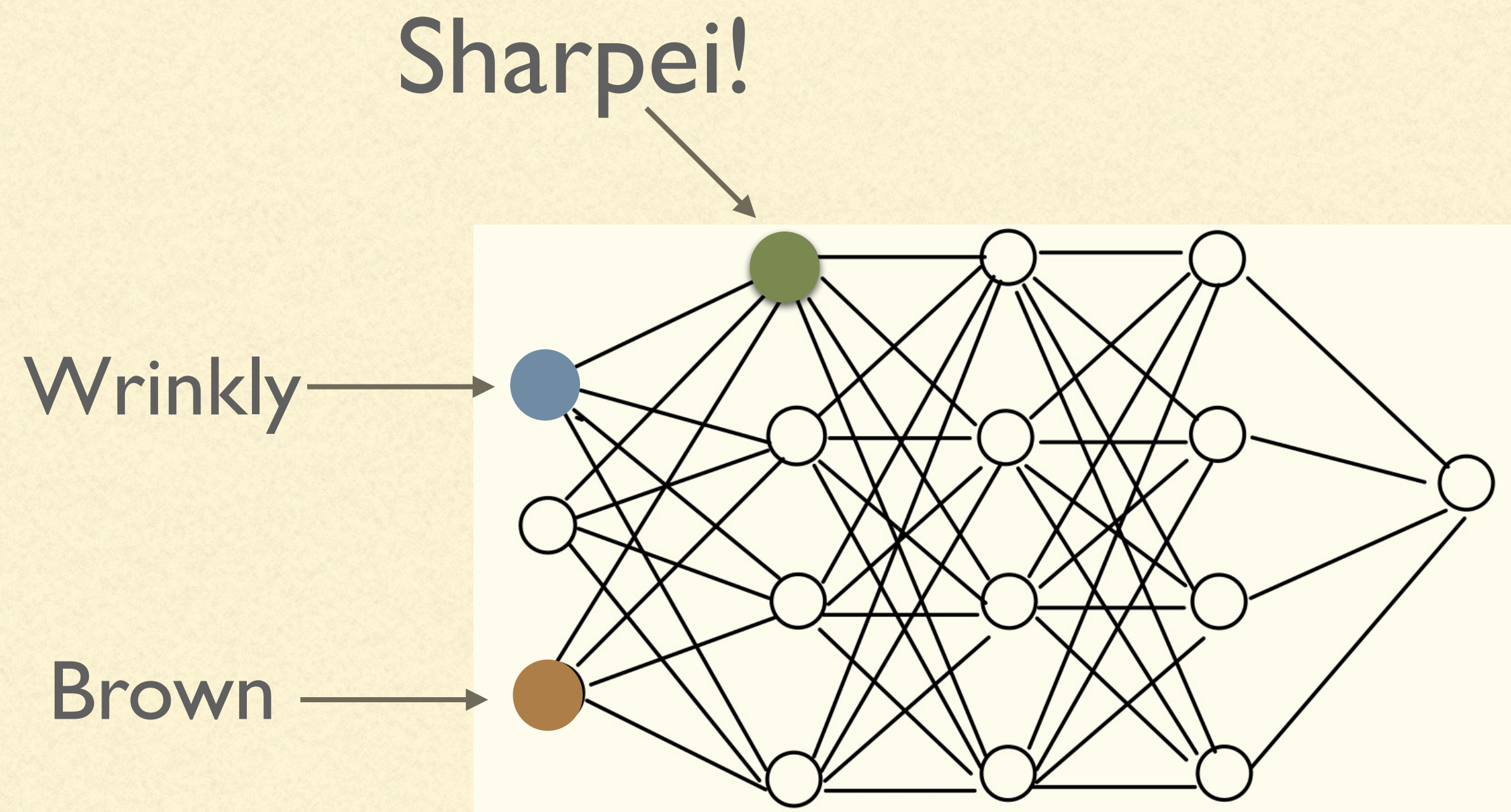


ALIGNMENT AND MONOSEMANTICITY:

(Anthropic interpretability team, Chris Olah, Neel Nanda, many many others)

Desirable situation for interpretability:

- *Alignment* of a model's computational units & human-interpretable concepts



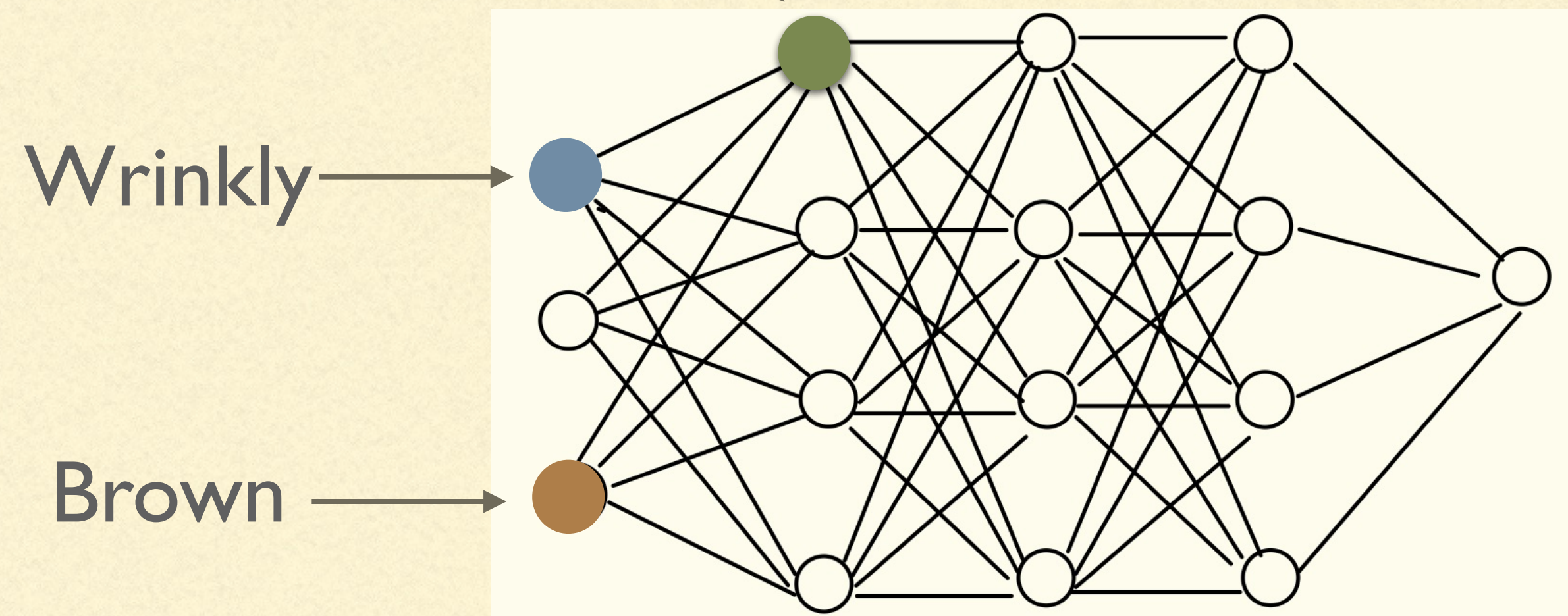
ALIGNMENT AND MONOSEMANTICITY:

(Anthropic interpretability team, Chris Olah, Neel Nanda, many many others)

Desirable situation for interpretability:

- *Alignment* of a model's computational units & human-interpretable concepts

Sharpei....or croissant????



Sharpei or croissant ??



ALIGNMENT AND MONOSEMANTICITY:

(Anthropic interpretability team, Chris Olah, Neel Nanda, many many others)

Desirable situation for interpretability:

- *Alignment* of a model's computational units & human-interpretable concepts
- Connectivity of a model's computational graph gives info about dependence of answer on concepts

ALIGNMENT AND MONOSEMANTICITY:

(Anthropic interpretability team, Chris Olah, Neel Nanda, many many others)

Desirable situation for interpretability:

- *Alignment* of a model's computational units & human-interpretable concepts
- Connectivity of a model's computational graph gives info about dependence of answer on concepts

Empirically, this doesn't happen:

ALIGNMENT AND MONOSEMANTICITY:

(Anthropic interpretability team, Chris Olah, Neel Nanda, many many others)

Desirable situation for interpretability:

- *Alignment* of a model's computational units & human-interpretable concepts
- Connectivity of a model's computational graph gives info about dependence of answer on concepts

Empirically, this doesn't happen:

- A single model can often handle questions about many different kinds of data, involving a number of different concepts (**Universality**)
-

ALIGNMENT AND MONOSEMANTICITY:

(Anthropic interpretability team, Chris Olah, Neel Nanda, many many others)

Desirable situation for interpretability:

- *Alignment* of a model's computational units & human-interpretable concepts
- Connectivity of a model's computational graph gives info about dependence of answer on concepts

Empirically, this doesn't happen:

- A single model can often handle questions about many different kinds of data, involving a number of different concepts (**Universality**)
 - Even after change-of-basis, there is no one-to-one correspondence between computational units and concepts
-

ALIGNMENT AND MONOSEMANTICITY:

(Anthropic interpretability team, Chris Olah, Neel Nanda, many many others)

Desirable situation for interpretability:

- *Alignment* of a model's computational units & human-interpretable concepts
- Connectivity of a model's computational graph gives info about dependence of answer on concepts

Empirically, this doesn't happen:

- A single model can often handle questions about many different kinds of data, involving a number of different concepts (**Universality**)
 - Even after change-of-basis, there is no one-to-one correspondence between computational units and concepts
 - *Still: only a few features in any data set seem to be important for every calculation* (**Sparsity**)
-

SUPERPOSITION HYPOTHESIS:

*Entangled
Model*

$$\mathbb{R}^{n_0} \xrightarrow{F^1} \mathbb{R}^{n_1} \xrightarrow{F^2} \dots \rightarrow \mathbb{R}^{n_{d-1}} \xrightarrow{F^d} \mathbb{R}^{n_d}$$

SUPERPOSITION HYPOTHESIS:

Disentangled
Model

$$\mathbb{R}^{N_0} \xrightarrow{\tilde{F}^1} \mathbb{R}^{N_1} \xrightarrow{\tilde{F}^2} \dots \rightarrow \mathbb{R}^{N_{d-1}} \xrightarrow{\tilde{F}^d} \mathbb{R}^{N_d}$$

Entangled
Model

$$\mathbb{R}^{n_0} \xrightarrow{F^1} \mathbb{R}^{n_1} \xrightarrow{F^2} \dots \rightarrow \mathbb{R}^{n_{d-1}} \xrightarrow{F^d} \mathbb{R}^{n_d}$$

SUPERPOSITION HYPOTHESIS:

*Disentangled
Model*

$$\mathbb{R}^{N_0} \xrightarrow{\tilde{F}^1} \mathbb{R}^{N_1} \xrightarrow{\tilde{F}^2} \dots \rightarrow \mathbb{R}^{N_{d-1}} \xrightarrow{\tilde{F}^d} \mathbb{R}^{N_d}$$

$$D_0 \left(\begin{array}{c} \nearrow \\ \searrow \end{array} \right) E_0 \quad D_1 \left(\begin{array}{c} \nearrow \\ \searrow \end{array} \right) E_1 \quad \dots \quad D_{d-1} \left(\begin{array}{c} \nearrow \\ \searrow \end{array} \right) E_{d-1} \quad D_d \left(\begin{array}{c} \nearrow \\ \searrow \end{array} \right) E_d$$

*Entangled
Model*

$$\mathbb{R}^{n_0} \xrightarrow{F^1} \mathbb{R}^{n_1} \xrightarrow{F^2} \dots \rightarrow \mathbb{R}^{n_{d-1}} \xrightarrow{F^d} \mathbb{R}^{n_d}$$

SUPERPOSITION HYPOTHESIS:

Privileged basis

$\leftrightarrow$

Interpretable
concepts

$$\mathbb{R}^{N_0} \xrightarrow{\tilde{F}^1} \mathbb{R}^{N_1} \xrightarrow{\tilde{F}^2} \dots \rightarrow \mathbb{R}^{N_{d-1}} \xrightarrow{\tilde{F}^d} \mathbb{R}^{N_d}$$

$$D_0 \left(\begin{array}{c} \nearrow \\ \searrow \end{array} \right) E_0 \quad D_1 \left(\begin{array}{c} \nearrow \\ \searrow \end{array} \right) E_1 \quad \dots \quad D_{d-1} \left(\begin{array}{c} \nearrow \\ \searrow \end{array} \right) E_{d-1} \quad D_d \left(\begin{array}{c} \nearrow \\ \searrow \end{array} \right) E_d$$

$$\mathbb{R}^{n_0} \xrightarrow{F^1} \mathbb{R}^{n_1} \xrightarrow{F^2} \dots \rightarrow \mathbb{R}^{n_{d-1}} \xrightarrow{F^d} \mathbb{R}^{n_d}$$

SUPERPOSITION HYPOTHESIS:

Privileged basis

$\leftrightarrow$

Interpretable
concepts

$$\mathbb{R}^{N_0} \xrightarrow{\tilde{F}^1} \mathbb{R}^{N_1} \xrightarrow{\tilde{F}^2} \dots \rightarrow \mathbb{R}^{N_{d-1}} \xrightarrow{\tilde{F}^d} \mathbb{R}^{N_d}$$

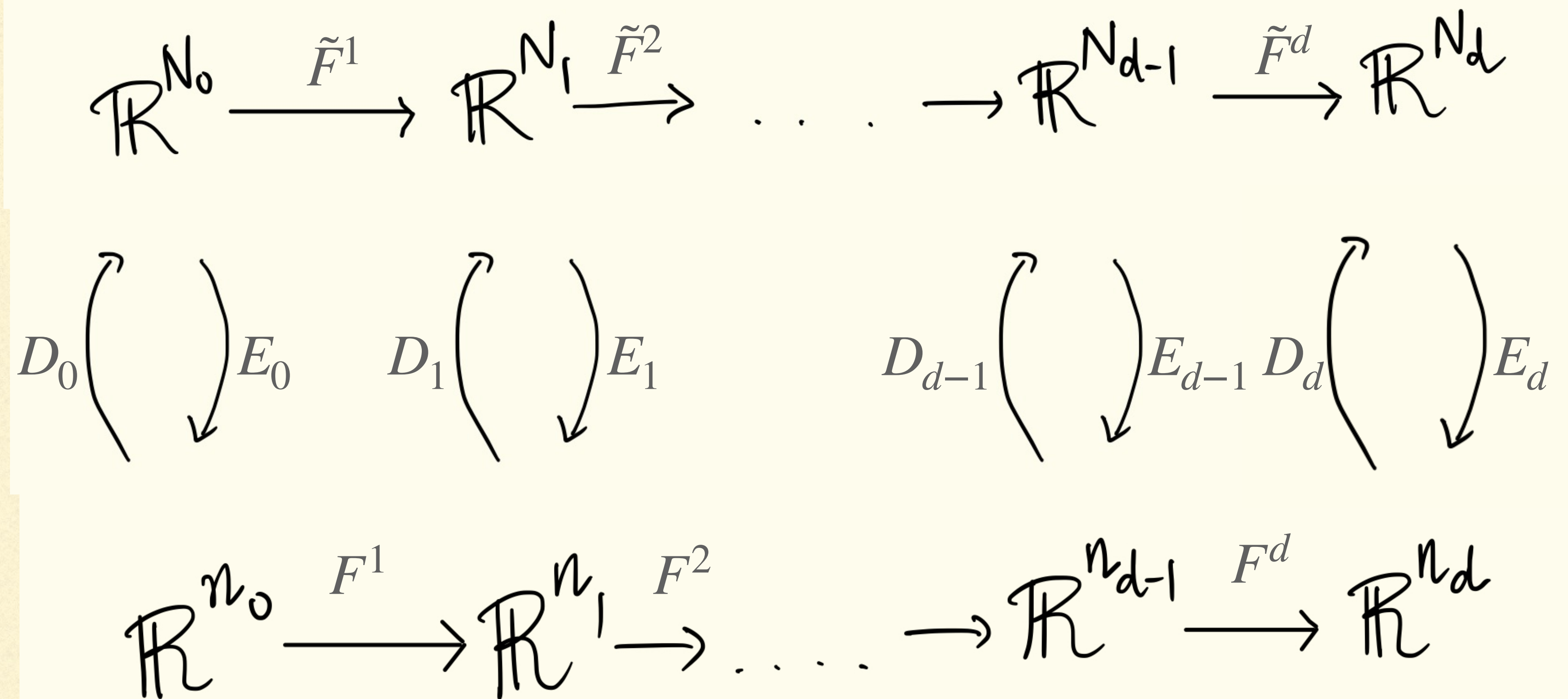
$$D_0 \left(\begin{array}{c} \nearrow \\ \searrow \end{array} \right) E_0 \quad D_1 \left(\begin{array}{c} \nearrow \\ \searrow \end{array} \right) E_1 \quad D_{d-1} \left(\begin{array}{c} \nearrow \\ \searrow \end{array} \right) E_{d-1} \quad D_d \left(\begin{array}{c} \nearrow \\ \searrow \end{array} \right) E_d$$

Images of privileged
basis are *nearly*
orthogonal

$$\mathbb{R}^{n_0} \xrightarrow{F^1} \mathbb{R}^{n_1} \xrightarrow{F^2} \dots \rightarrow \mathbb{R}^{n_{d-1}} \xrightarrow{F^d} \mathbb{R}^{n_d}$$

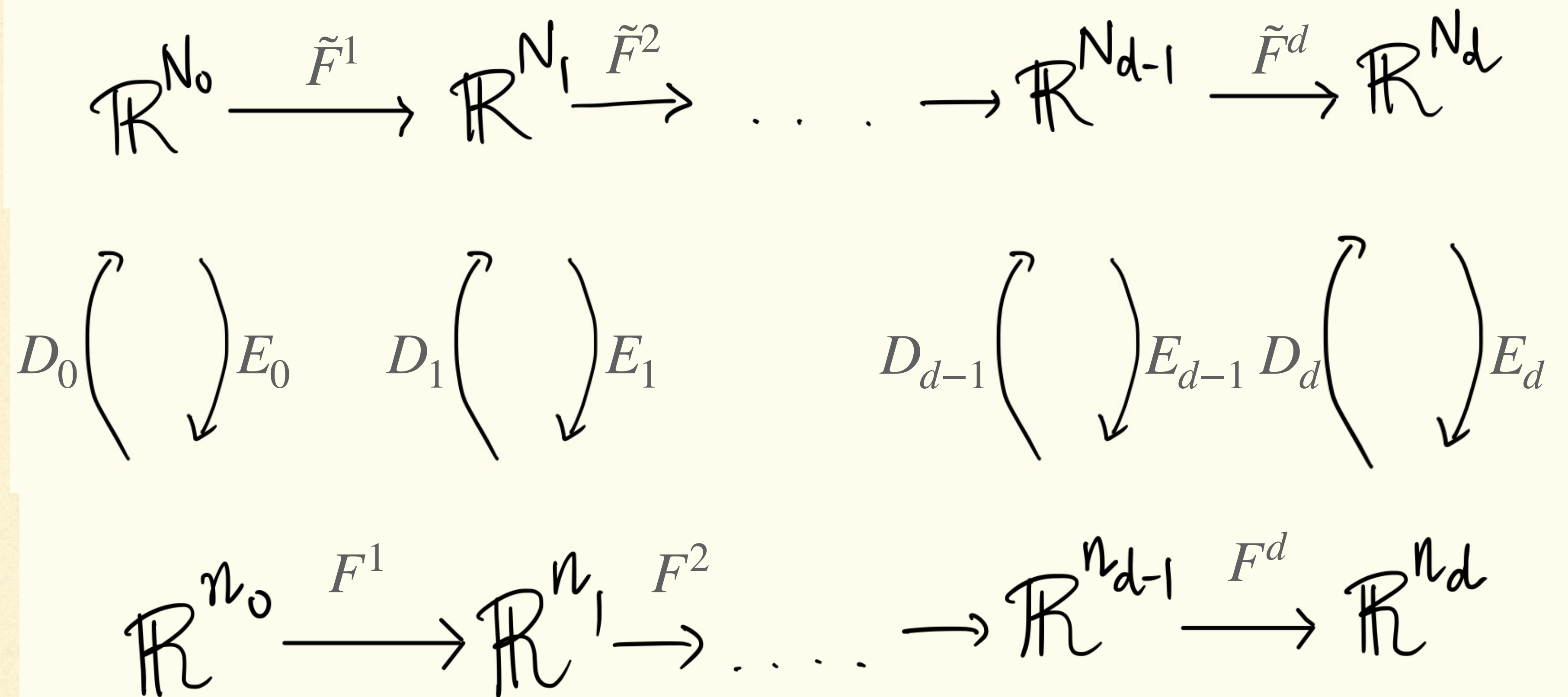
SUPERPOSITION HYPOTHESIS:

Lift to disentangled
model is *sparse* on
real world data sets
(only a few
concepts are
actively involved in
any given
computation)



SUPERPOSITION HYPOTHESIS:

Strength of activation
on a data set in
disentangled model
corresponds to
strength of the
corresponding
concept in that data
set



SUPERPOSITION HYPOTHESIS:

To understand a computation in the *entangled model*, use the dictionary from the disentangled model

$$\begin{array}{ccccccc} \mathbb{R}^{N_0} & \xrightarrow{\tilde{F}^1} & \mathbb{R}^{N_1} & \xrightarrow{\tilde{F}^2} & \dots & \rightarrow & \mathbb{R}^{N_{d-1}} & \xrightarrow{\tilde{F}^d} & \mathbb{R}^{N_d} \\ \\ D_0 \left(\begin{array}{c} \nearrow \\ \searrow \end{array} \right) E_0 & & D_1 \left(\begin{array}{c} \nearrow \\ \searrow \end{array} \right) E_1 & & & & D_{d-1} \left(\begin{array}{c} \nearrow \\ \searrow \end{array} \right) E_{d-1} & & D_d \left(\begin{array}{c} \nearrow \\ \searrow \end{array} \right) E_d \\ \\ \mathbb{R}^{n_0} & \xrightarrow{F^1} & \mathbb{R}^{n_1} & \xrightarrow{F^2} & \dots & \rightarrow & \mathbb{R}^{n_{d-1}} & \xrightarrow{F^d} & \mathbb{R}^{n_d} \end{array}$$

FINDING CIRCUITS, TRANSFORMER LENS LIBRARY:

- Attribution (Like saliency analysis)
- Activation patching (Causal intervention)
- Ablation (Causal intervention)

In principle, would allow us to pare down the calculation into something simple and interpretable

OUTLINE:

1. Supervised learning using neural network models:
Sequence processing
 2. Mechanistic interpretability: key structural ideas and
empirical methods
 3. Case study: the word problem in Coxeter groups
-

BC EXPERIMENTAL MATH & ML LAB

BC EXPERIMENTAL MATH & ML LAB

Project Team



Caleb Ashley
Co-director



Vanessa Rigoglioso



Roman Roy



David Ruiz De Castilla



Rob Vollbrecht



Gina Wang



Celina Xiong

WORD PROBLEM IN (FINITELY PRESENTED) GROUPS:

Given w and w' two *words* in generators and their inverses, do they represent the same element of the group?

Equivalently, does $w^{-1}w' = e$, the identity element of the group?

WORD PROBLEM IN (FINITELY PRESENTED) GROUPS:

Given w and w' two words in generators and their inverses, do they represent the same element of the group?

Equivalently, does $w^{-1}w' = e$, the identity element of the group?

For general finitely presented groups: no algorithm for solving the word problem

WORD PROBLEM IN (FINITELY PRESENTED) GROUPS:

Given w and w' two words in generators and their inverses, do they represent the same element of the group?

Equivalently, does $w^{-1}w' = e$, the identity element of the group?

For general finitely presented groups: no algorithm for solving the word problem

For Coxeter groups: there are several efficient algorithms, all best understood geometrically

COXETER GROUPS:

Presentations: $\langle s_1, \dots, s_n \mid (s_i s_j)^{m_{ij}} \rangle$

$$\begin{aligned} m_{ij} &= 1 \text{ if } i = j \\ m_{ij} &\in \mathbb{Z}^{\geq 2} \cup \{\infty\} \\ &\text{if } i \neq j \end{aligned}$$

COXETER GROUPS:

Presentations: $\langle s_1, \dots, s_n \mid (s_i s_j)^{m_{ij}} \rangle$

$$\begin{aligned} m_{ij} &= 1 \text{ if } i = j \\ m_{ij} &\in \mathbb{Z}^{\geq 2} \cup \{\infty\} \\ &\text{if } i \neq j \end{aligned}$$

Canonical Geometric Realization:

COXETER GROUPS:

Presentations: $\langle s_1, \dots, s_n \mid (s_i s_j)^{m_{ij}} \rangle$

$$\begin{aligned} m_{ij} &= 1 \text{ if } i = j \\ m_{ij} &\in \mathbb{Z}^{\geq 2} \cup \{\infty\} \\ &\text{if } i \neq j \end{aligned}$$

Canonical Geometric Realization:

- Generators act via reflections across real hyperplanes: $s_i^2 = e$ for all i
-

COXETER GROUPS:

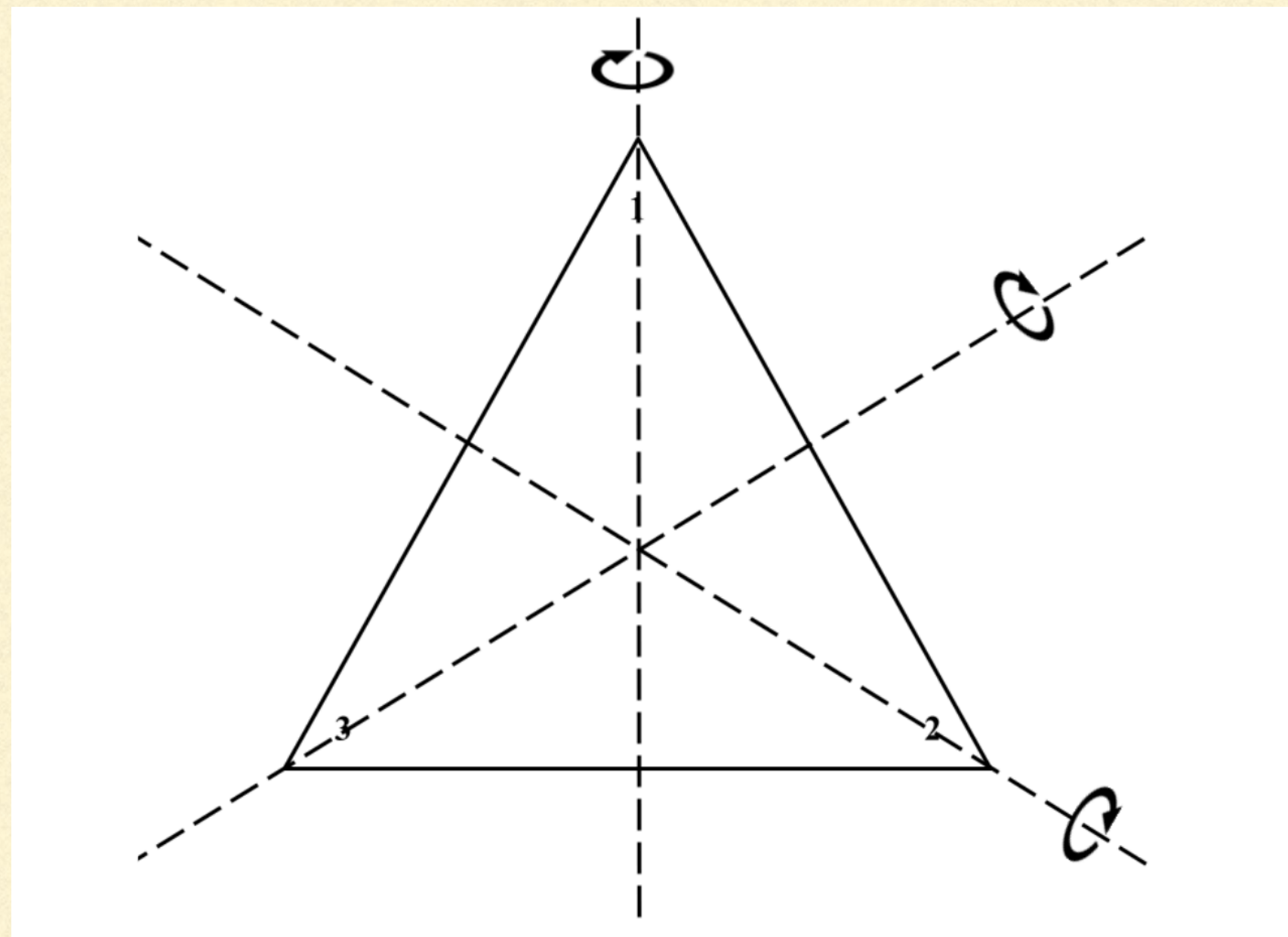
Presentations: $\langle s_1, \dots, s_n \mid (s_i s_j)^{m_{ij}} \rangle$

$$\begin{aligned} m_{ij} &= 1 \text{ if } i = j \\ m_{ij} &\in \mathbb{Z}^{\geq 2} \cup \{\infty\} \\ &\text{if } i \neq j \end{aligned}$$

Canonical Geometric Realization:

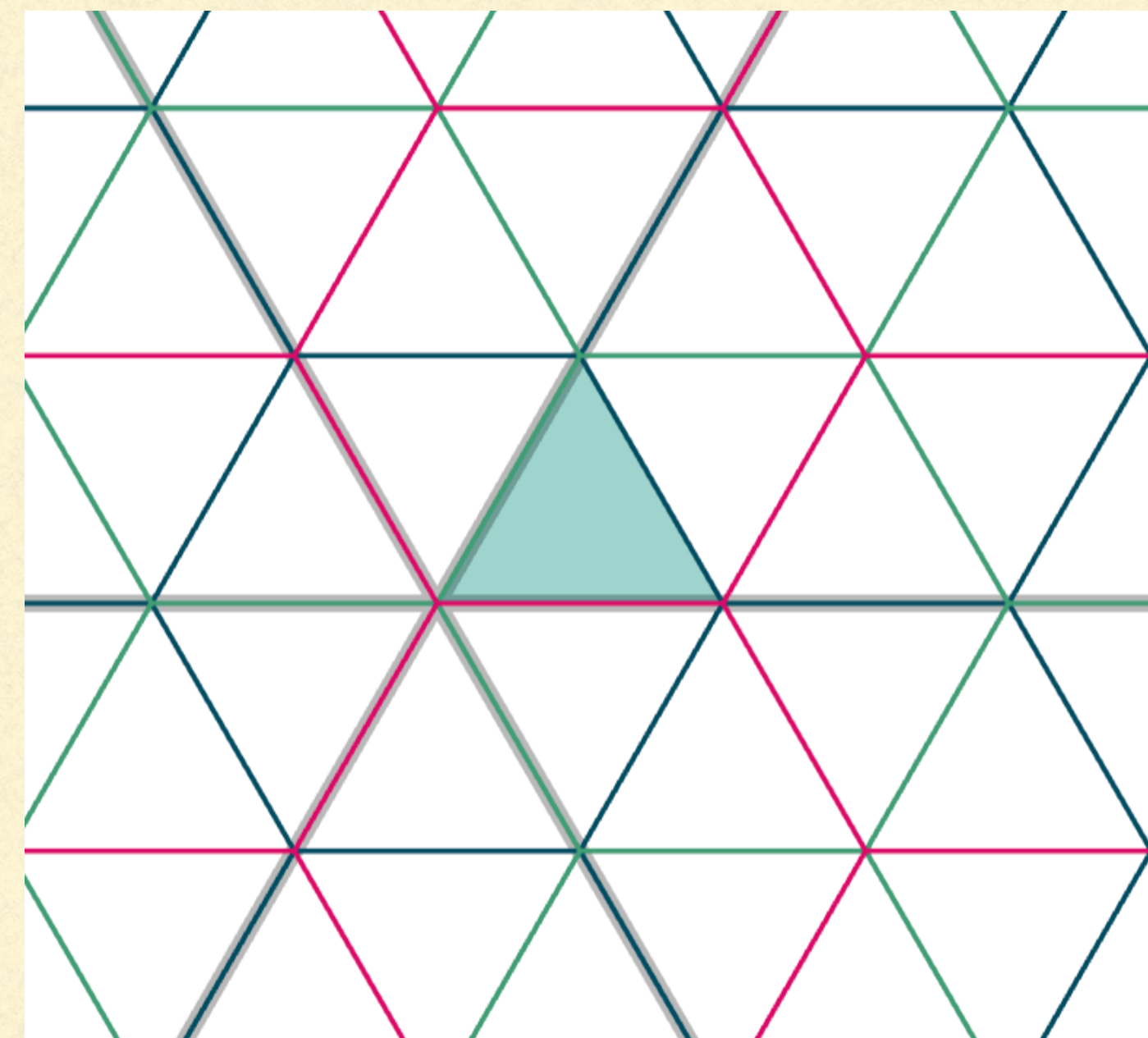
- Generators act via reflections across real hyperplanes: $s_i^2 = e$ for all i
 - “Braiding relations” $(s_i s_j)^{m_{ij}}$ for $i \neq j$ encode how the hyperplanes intersect
-

SIMPLEST EXAMPLES (FINITE & INFINITE):



S_3

Image credit: BCExpMMLL team



$\tilde{A}_2$

Image credit: Joel Gibson, *LieVis*

REPRESENTATION-THEORETIC SOLUTION(S):

A Toy Model of Universality: Reverse Engineering how Networks Learn Group Operations

Bilal Chughtai¹ Lawrence Chan² Neel Nanda¹

Abstract

Universality is a key hypothesis in mechanistic interpretability – that different models learn similar features and circuits when trained on similar tasks. In this work, we study the universality hypothesis by examining how small neural networks learn to implement group composition. We present a novel algorithm by which neural networks may implement composition for any finite group via mathematical representation theory. We then show that networks consistently learn this algorithm by reverse engineering model logits and weights, and confirm our understanding using ablations. By studying networks of differing architectures trained on various groups, we find mixed evidence for universality: using our algorithm, we can completely characterize the family of circuits and features that networks learn on this task, but for a given network the precise circuits learned – as well as the order they develop – are arbitrary.

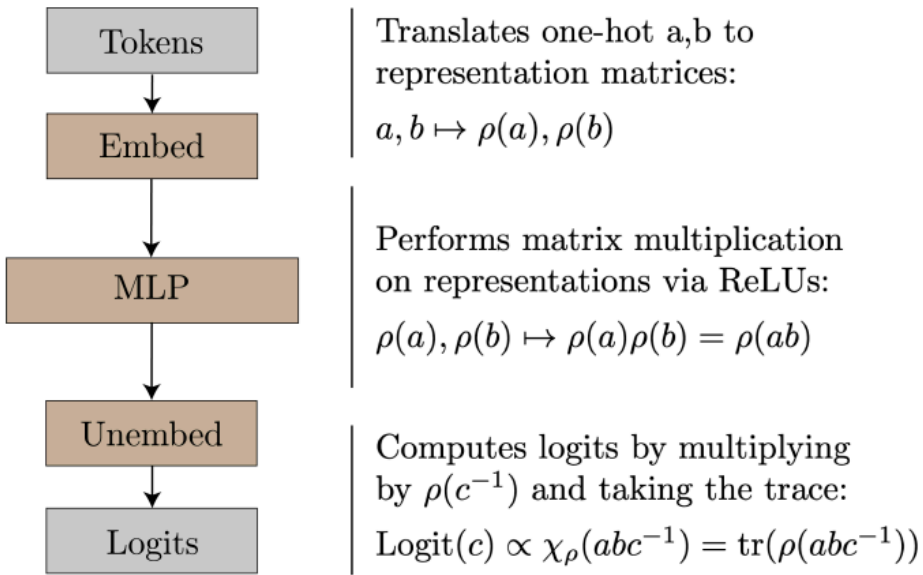


Figure 1. The algorithm implemented by a one hidden layer MLP for arbitrary group composition. Given two input group elements a and b , the model learns representation matrices $\rho(a)$ and $\rho(b)$ in its embeddings. Using the ReLU activations in its MLP layer, it then multiplies these matrices, computing $\rho(a)\rho(b) = \rho(ab)$. Finally, it ‘reads off’ the logits for each output group element c by computing *characters* – the matrix trace $\text{tr} \rho(abc^{-1})$, denoted $\chi_\rho(abc^{-1})$, which is maximized when $c = ab$.

1. Introduction

Do models converge on the same solutions to a task, or are the algorithms implemented arbitrary and unpredictable? The *universality hypothesis* (Olah et al., 2020; Li et al., 2016) asserts that models learn similar features and cir-

and circuits in detail (Elhage et al., 2021; Wang et al., 2022; Nanda et al., 2023). If the universality hypothesis holds, then the insights and principles found by studying small models will transfer to state-of-the-art models that are used in practice. But if universality is false, then although we

REPRESENTATION-THEORETIC SOLUTION(S):

A Toy Model of Universality: Reverse Engineering how Networks Learn Group Operations

Bilal Chughtai¹ Lawrence Chan² Neel Nanda¹

Abstract

Universality is a key hypothesis in mechanistic interpretability – that different models learn similar features and circuits when trained on similar tasks. In this work, we study the universality hypothesis by examining how small neural networks learn to implement group composition. We present a novel algorithm by which neural networks may implement composition for any finite group via mathematical representation theory. We then show that networks consistently learn this algorithm by reverse engineering model logits and weights, and confirm our understanding using ablations. By studying networks of differing architectures trained on various groups, we find mixed evidence for universality: using our algorithm, we can completely characterize the family of circuits and features that networks learn on this task, but for a given network the precise circuits learned – as well as the order they develop – are arbitrary.

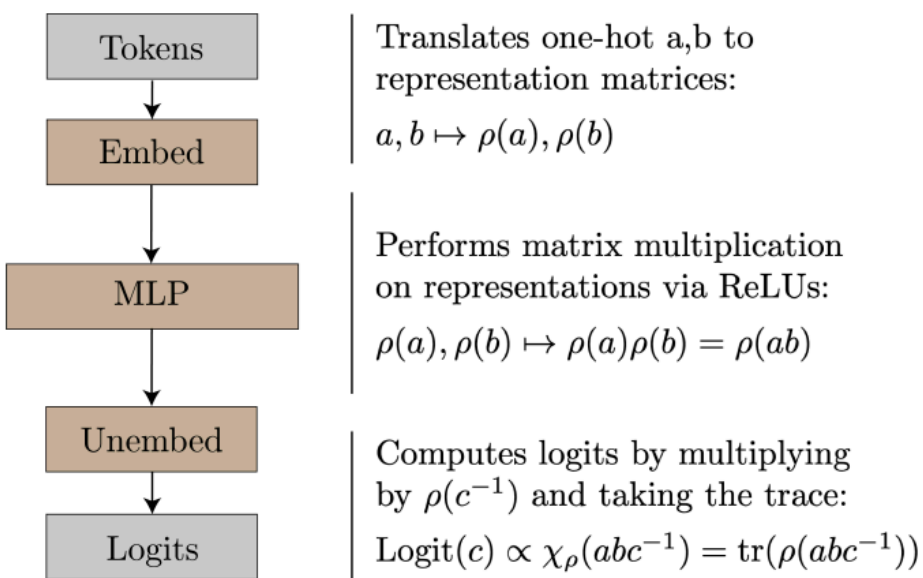


Figure 1. The algorithm implemented by a one hidden layer MLP for arbitrary group composition. Given two input group elements a and b , the model learns representation matrices $\rho(a)$ and $\rho(b)$ in its embeddings. Using the ReLU activations in its MLP layer, it then multiplies these matrices, computing $\rho(a)\rho(b) = \rho(ab)$. Finally, it ‘reads off’ the logits for each output group element c by computing *characters* – the matrix trace $\text{tr} \rho(abc^{-1})$, denoted $\chi_\rho(abc^{-1})$, which is maximized when $c = ab$.

1. Introduction

Do models converge on the same solutions to a task, or are the algorithms implemented arbitrary and unpredictable? The *universality hypothesis* (Olah et al., 2020; Li et al., 2016) asserts that models learn similar features and cir-

and circuits in detail (Elhage et al., 2021; Wang et al., 2022; Nanda et al., 2023). If the universality hypothesis holds, then the insights and principles found by studying small models will transfer to state-of-the-art models that are used in practice. But if universality is false, then although we

Focuses on training transformers to learn how to multiply in some finite nonabelian groups, and inferring how the model does it

REPRESENTATION-THEORETIC SOLUTION(S):

A Toy Model of Universality: Reverse Engineering how Networks Learn Group Operations

Bilal Chughtai¹ Lawrence Chan² Neel Nanda¹

Abstract

Universality is a key hypothesis in mechanistic interpretability – that different models learn similar features and circuits when trained on similar tasks. In this work, we study the universality hypothesis by examining how small neural networks learn to implement group composition. We present a novel algorithm by which neural networks may implement composition for any finite group via mathematical representation theory. We then show that networks consistently learn this algorithm by reverse engineering model logits and weights, and confirm our understanding using ablations. By studying networks of differing architectures trained on various groups, we find mixed evidence for universality: using our algorithm, we can completely characterize the family of circuits and features that networks learn on this task, but for a given network the precise circuits learned – as well as the order they develop – are arbitrary.

1. Introduction

Do models converge on the same solutions to a task, or are the algorithms implemented arbitrary and unpredictable? The *universality hypothesis* (Olah et al., 2020; Li et al., 2016) asserts that models learn similar features and cir-

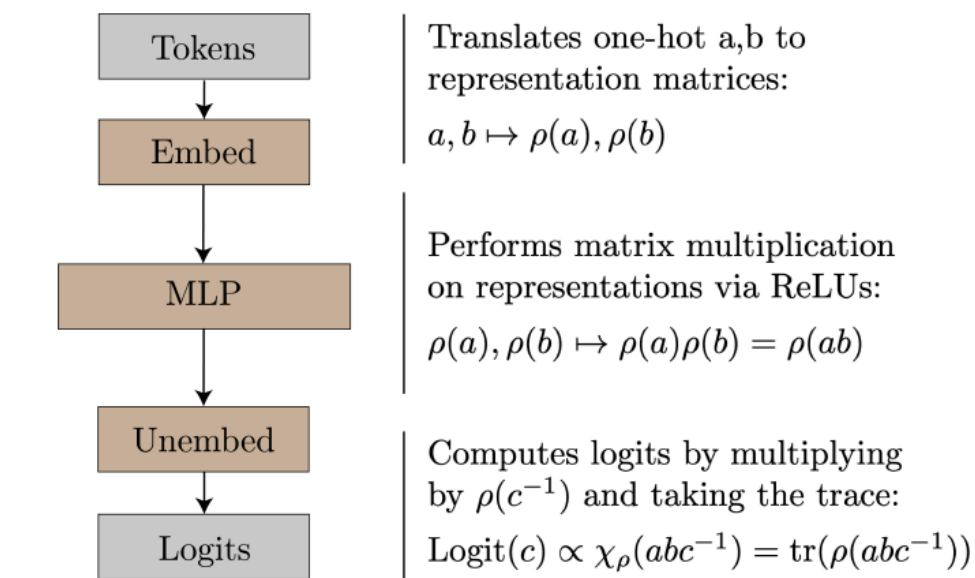


Figure 1. The algorithm implemented by a one hidden layer MLP for arbitrary group composition. Given two input group elements a and b , the model learns representation matrices $\rho(a)$ and $\rho(b)$ in its embeddings. Using the ReLU activations in its MLP layer, it then multiplies these matrices, computing $\rho(a)\rho(b) = \rho(ab)$. Finally, it ‘reads off’ the logits for each output group element c by computing *characters* – the matrix trace $\text{tr} \rho(abc^{-1})$, denoted $\chi_\rho(abc^{-1})$, which is maximized when $c = ab$.

and circuits in detail (Elhage et al., 2021; Wang et al., 2022; Nanda et al., 2023). If the universality hypothesis holds, then the insights and principles found by studying small models will transfer to state-of-the-art models that are used in practice. But if universality is false, then although we

Learn a faithful linear representation?

Focuses on training transformers to learn how to multiply in some finite nonabelian groups, and inferring how the model does it

REPRESENTATION-THEORETIC SOLUTION(S):

For a **faithful representation** of a **finite** group G , trace
detects the identity element

REPRESENTATION-THEORETIC SOLUTION(S):

For a **faithful representation** of a **finite** group G , trace
is uniquely maximized on the identity element

REPRESENTATION-THEORETIC SOLUTION(S):

Lemma: If G is a **finite** group, $\rho : G \rightarrow GL(V)$ an n -dimensional faithful representation,
 $\chi_\rho : G \rightarrow \mathbb{R}$ its associated character $\chi_\rho(g) := \text{trace}(\rho(g))$,
 $\chi_\rho(g) = n$ iff $g = e$.

REPRESENTATION-THEORETIC SOLUTION(S):

Lemma: If G is a **finite** group, $\rho : G \rightarrow GL(V)$ an n -dimensional faithful representation, $\chi_\rho : G \rightarrow \mathbb{R}$ its associated character $\chi_\rho(g) := \text{trace}(\rho(g))$,
$$\chi_\rho(g) = n \text{ iff } g = e.$$

Proof:

Since all $g \in G$ have finite order, eigenvalues $\lambda_1, \dots, \lambda_n \in \mathbb{C}$ of $\rho(g)$ are roots of unity.

REPRESENTATION-THEORETIC SOLUTION(S):

Lemma: If G is a **finite** group, $\rho : G \rightarrow GL(V)$ an n -dimensional faithful representation, $\chi_\rho : G \rightarrow \mathbb{R}$ its associated character $\chi_\rho(g) := \text{trace}(\rho(g))$,
 $\chi_\rho(g) = n$ iff $g = e$.

Proof:

Since all $g \in G$ have finite order, eigenvalues $\lambda_1, \dots, \lambda_n \in \mathbb{C}$ of $\rho(g)$ are roots of unity.

$|\lambda_1 + \dots + \lambda_n| \leq |\lambda_1| + \dots + |\lambda_n| = n$, with equality iff $\lambda_i = 1$ for all i .

REPRESENTATION-THEORETIC SOLUTION(S):

Lemma: If G is a **finite** group, $\rho : G \rightarrow GL(V)$ an n -dimensional faithful representation, $\chi_\rho : G \rightarrow \mathbb{R}$ its associated character $\chi_\rho(g) := \text{trace}(\rho(g))$,
$$\chi_\rho(g) = n \text{ iff } g = e.$$

Proof:

Since all $g \in G$ have finite order, eigenvalues $\lambda_1, \dots, \lambda_n \in \mathbb{C}$ of $\rho(g)$ are roots of unity.

$|\lambda_1 + \dots + \lambda_n| \leq |\lambda_1| + \dots + |\lambda_n| = n$, with equality iff $\lambda_i = 1$ for all i .

Maschke's theorem (Finite dimensional reps of finite groups are semisimple) implies that if all eigenvalues are 1 then $\rho(g) = \text{Id}$.

REPRESENTATION-THEORETIC SOLUTION(S):

A Toy Model of Universality: Reverse Engineering how Networks Learn Group Operations

Bilal Chughtai¹ Lawrence Chan² Neel Nanda¹

Abstract

Universality is a key hypothesis in mechanistic interpretability – that different models learn similar features and circuits when trained on similar tasks. In this work, we study the universality hypothesis by examining how small neural networks learn to implement group composition. We present a novel algorithm by which neural networks may implement composition for any finite group via mathematical representation theory. We then show that networks consistently learn this algorithm by reverse engineering model logits and weights, and confirm our understanding using ablations. By studying networks of differing architectures trained on various groups, we find mixed evidence for universality: using our algorithm, we can completely characterize the family of circuits and features that networks learn on this task, but for a given network the precise circuits learned – as well as the order they develop – are arbitrary.

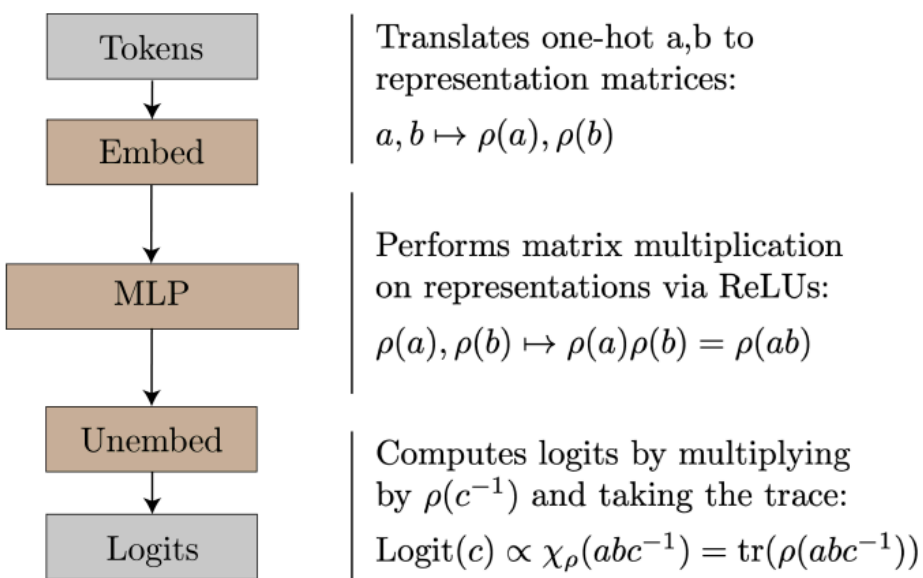


Figure 1. The algorithm implemented by a one hidden layer MLP for arbitrary group composition. Given two input group elements a and b , the model learns representation matrices $\rho(a)$ and $\rho(b)$ in its embeddings. Using the ReLU activations in its MLP layer, it then multiplies these matrices, computing $\rho(a)\rho(b) = \rho(ab)$. Finally, it ‘reads off’ the logits for each output group element c by computing *characters* – the matrix trace $\text{tr} \rho(abc^{-1})$, denoted $\chi_\rho(abc^{-1})$, which is maximized when $c = ab$.

1. Introduction

Do models converge on the same solutions to a task, or are the algorithms implemented arbitrary and unpredictable? The *universality hypothesis* (Olah et al., 2020; Li et al., 2016) asserts that models learn similar features and cir-

and circuits in detail (Elhage et al., 2021; Wang et al., 2022; Nanda et al., 2023). If the universality hypothesis holds, then the insights and principles found by studying small models will transfer to state-of-the-art models that are used in practice. But if universality is false, then although we

By training a small transformer model on group multiplication data for various finite groups and probing the trained model using transformerlens, they build evidence that the transformer is learning to multiply group elements by learning faithful representations of the group(s)

AUTOMATA FOR COXETER GROUPS:

AUTOMATA FOR COXETER GROUPS:

A finitely-presented group is said to be *(bi)automatic* if:

AUTOMATA FOR COXETER GROUPS:

A finitely-presented group is said to be *(bi)automatic* if:

- There exists a set of preferred words (*regular language* in generators and their inverses) in bijective correspondence with group elements

AUTOMATA FOR COXETER GROUPS:

A finitely-presented group is said to be *(bi)automatic* if:

- There exists a set of preferred words (*regular language* in generators and their inverses) in bijective correspondence with group elements
- There exists a finite state machine (*recognizing automaton*) recognizing whether a given word is in the regular language (is a *regular expression*)

AUTOMATA FOR COXETER GROUPS:

A finitely-presented group is said to be *(bi)automatic* if:

- There exists a set of preferred words (*regular language* in generators and their inverses) in bijective correspondence with group elements
- There exists a finite state machine (*recognizing automaton*) recognizing whether a given word is in the regular language (is a *regular expression*)
- There exists a finite state machine (*multiplication automaton*) taking a regular expression w and a generator s_i and finding the unique regular expression for ws_i

AUTOMATA FOR COXETER GROUPS:

A finitely-presented group is said to be *(bi)automatic* if:

- There exists a set of preferred words (*regular language* in generators and their inverses) in bijective correspondence with group elements
- There exists a finite state machine (*recognizing automaton*) recognizing whether a given word is in the regular language (is a *regular expression*)
- There exists a finite state machine (*multiplication automaton*) taking a regular expression w and a generator s_i and finding the unique regular expression for ws_i

Coxeter groups-even infinite ones-are automatic

AUTOMATA FOR COXETER GROUPS:

A finitely-presented group is said to be *(bi)automatic* if:

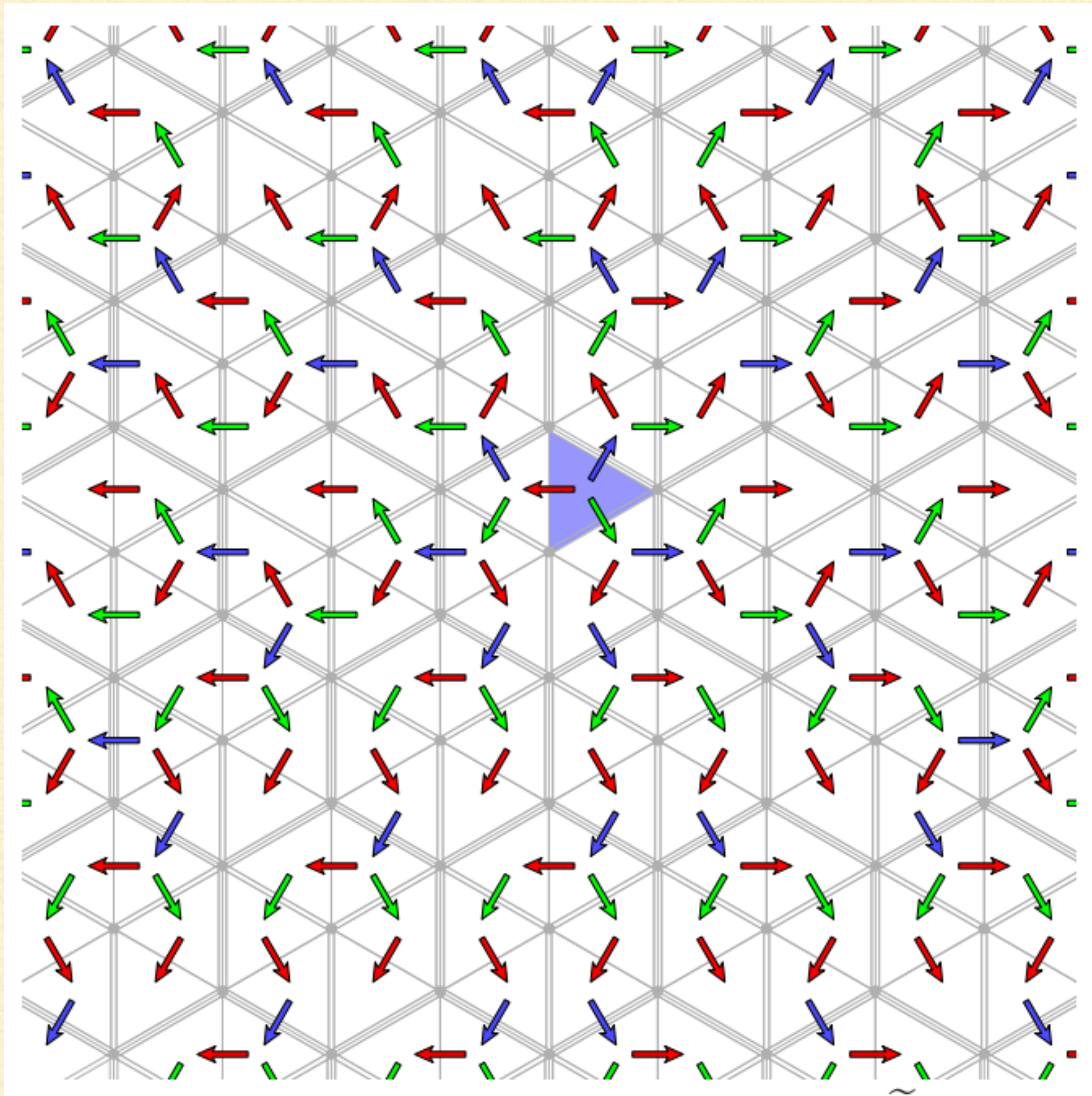
- There exists a set of preferred words (*regular language* in generators and their inverses) in bijective correspondence with group elements
- There exists a finite state machine (*recognizing automaton*) recognizing whether a given word is in the regular language (is a *regular expression*)
- There exists a finite state machine (*multiplication automaton*) taking a regular expression w and a generator s_i and finding the unique regular expression for ws_i

Coxeter groups-even infinite ones-are automatic

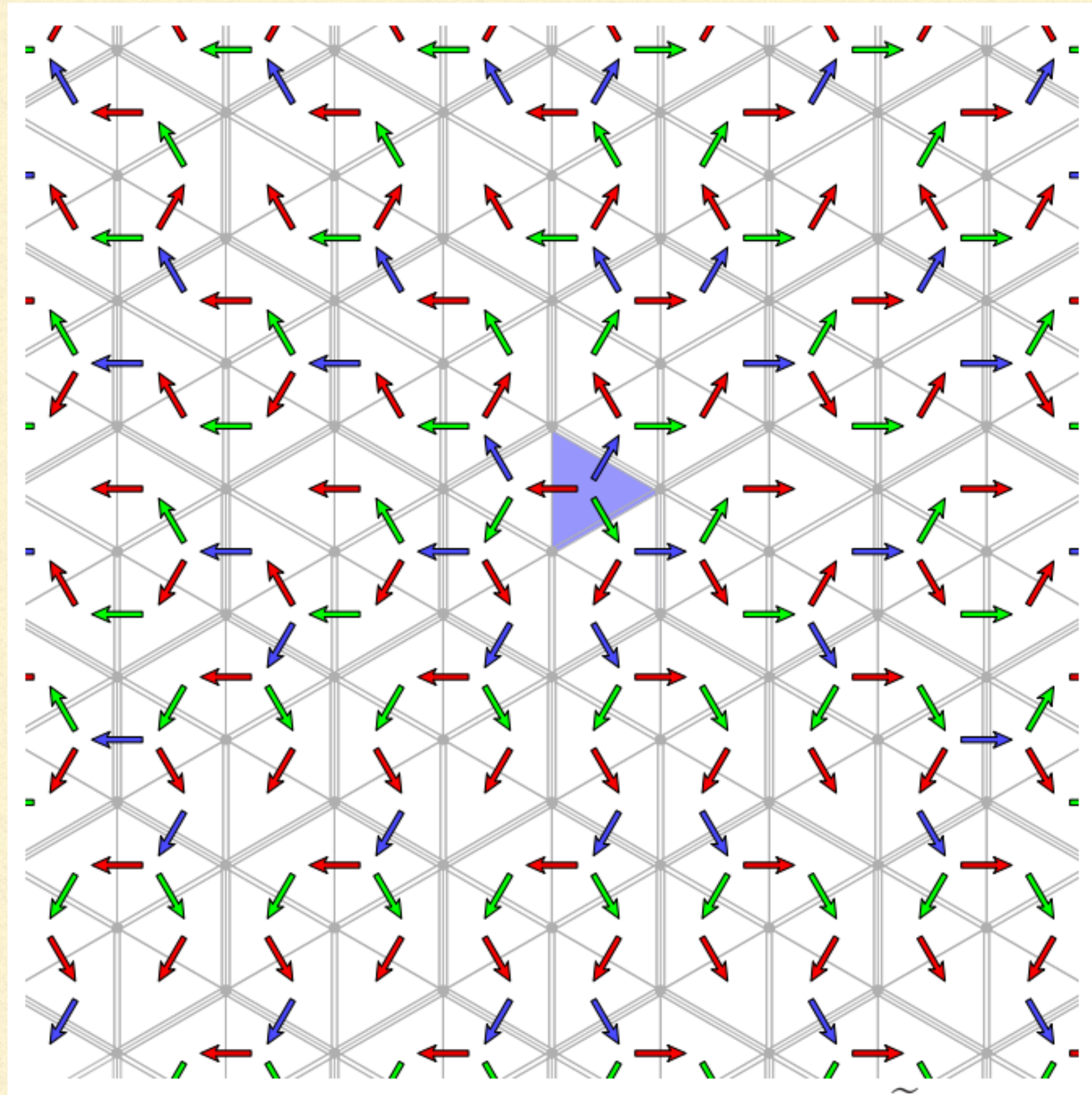
The recognizing and multiplication automata are geometric

Hat tip: Tony Licata

AUTOMATA FOR COXETER GROUPS:



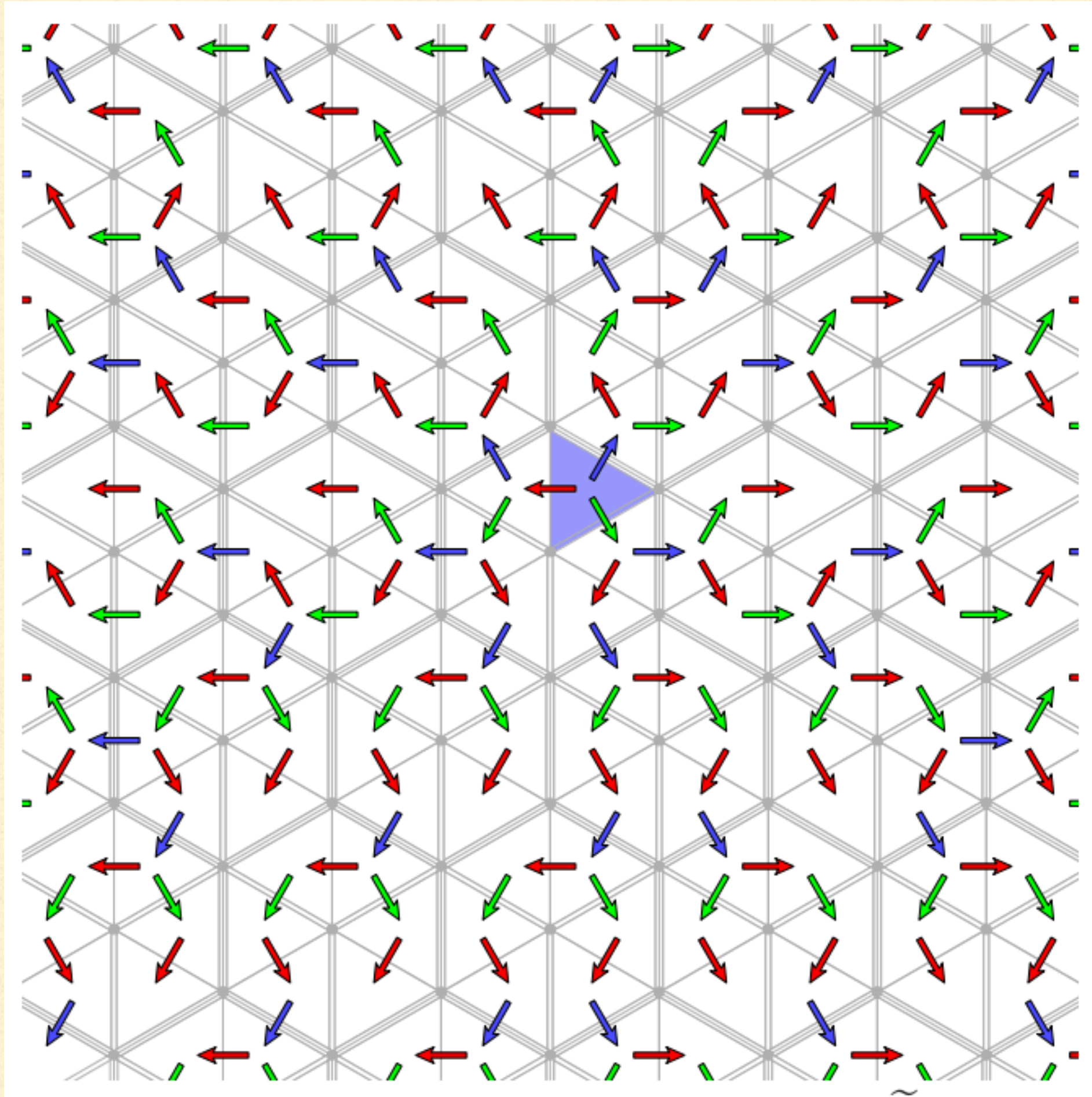
AUTOMATA FOR COXETER GROUPS:



Inverse Short Lex Regular Language:

Among shortest words representing a group element, choose the one that is least in reverse lexicographic order ($s_R < s_G < s_B$)

AUTOMATA FOR COXETER GROUPS:

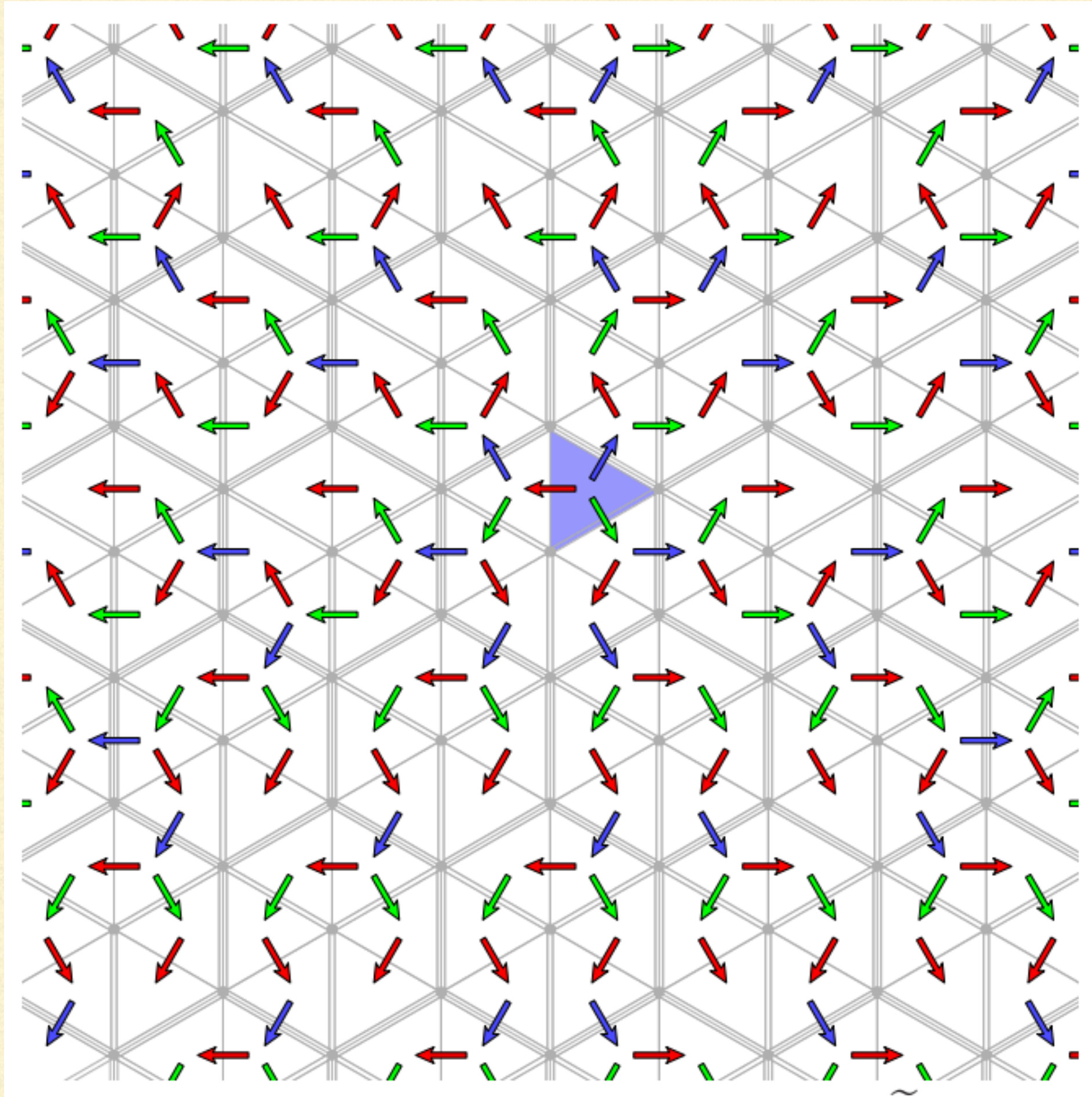


Inverse Short Lex Regular Language:

Among shortest words representing a group element, choose the one that is least in reverse lexicographic order ($s_R < s_G < s_B$)

- Invariant slice of (3D) *canonical geometric realization* of $\tilde{A}_2$

AUTOMATA FOR COXETER GROUPS:

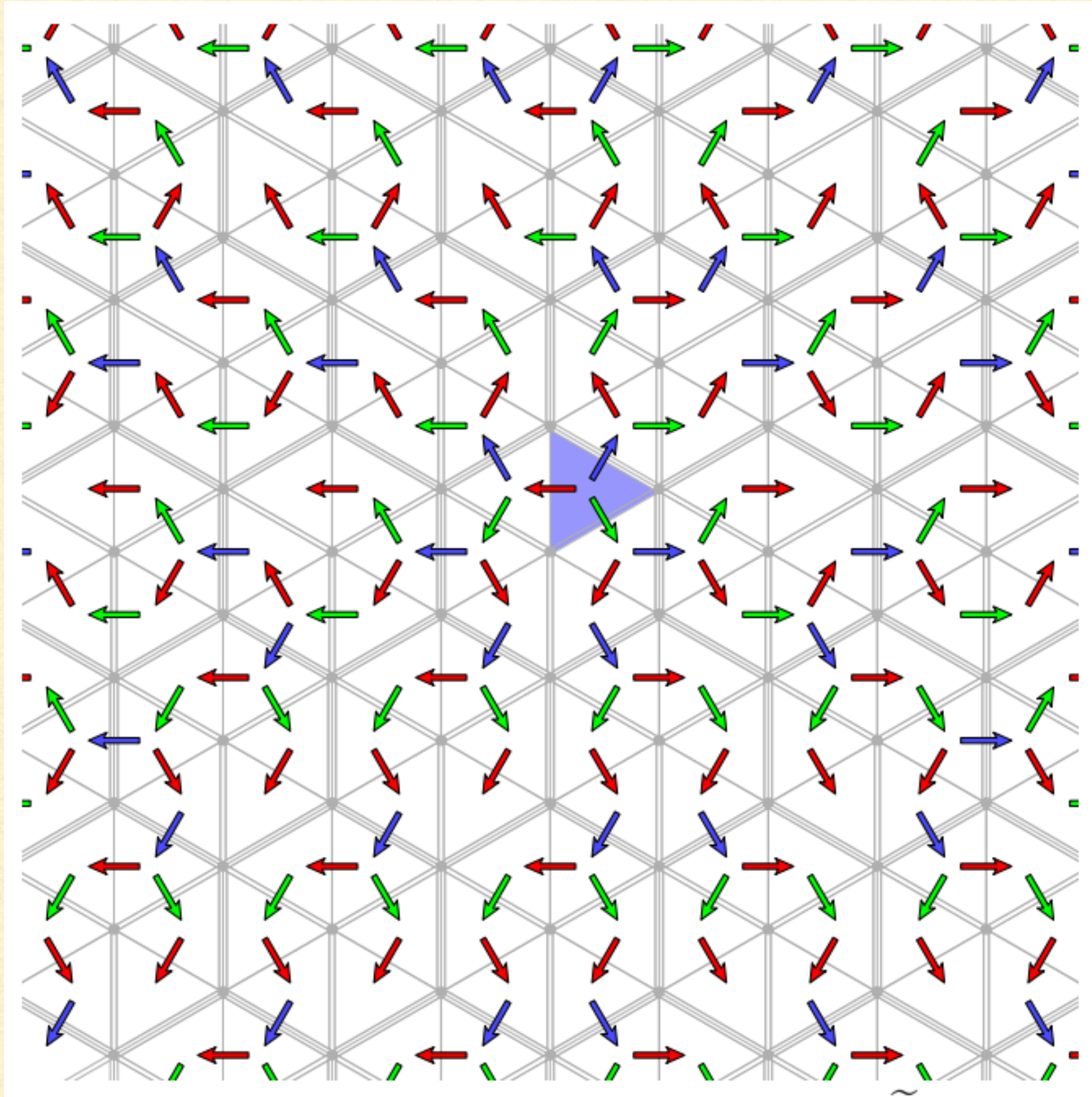


Inverse Short Lex Regular Language:

Among shortest words representing a group element, choose the one that is least in reverse lexicographic order ($s_R < s_G < s_B$)

- Invariant slice of (3D) *canonical geometric realization* of $\tilde{A}_2$
- Dual graph of tiling is Cayley graph of $\tilde{A}_2$

AUTOMATA FOR COXETER GROUPS:

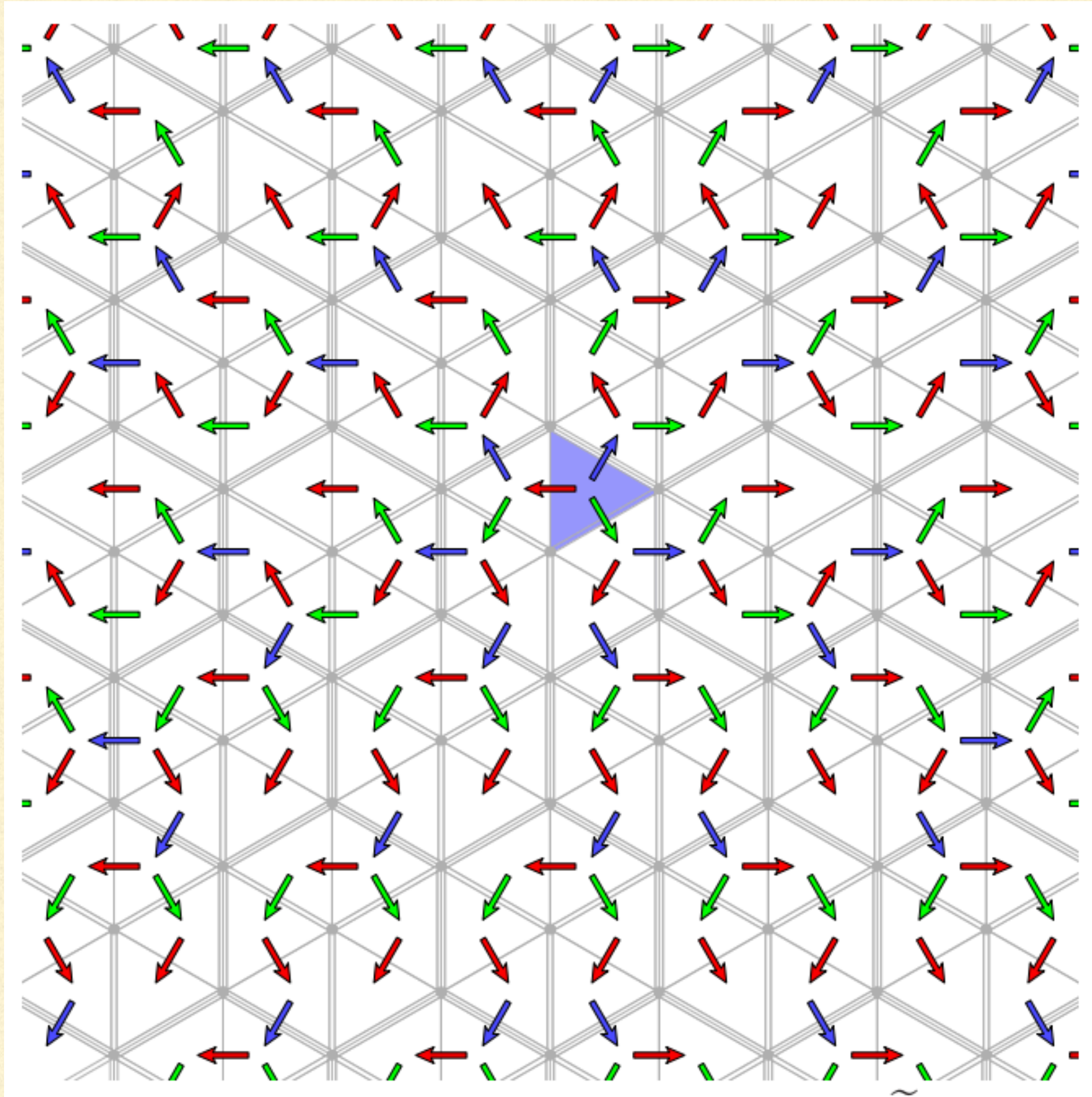


Inverse Short Lex Regular Language:

Among shortest words representing a group element, choose the one that is least in reverse lexicographic order ($s_R < s_G < s_B$)

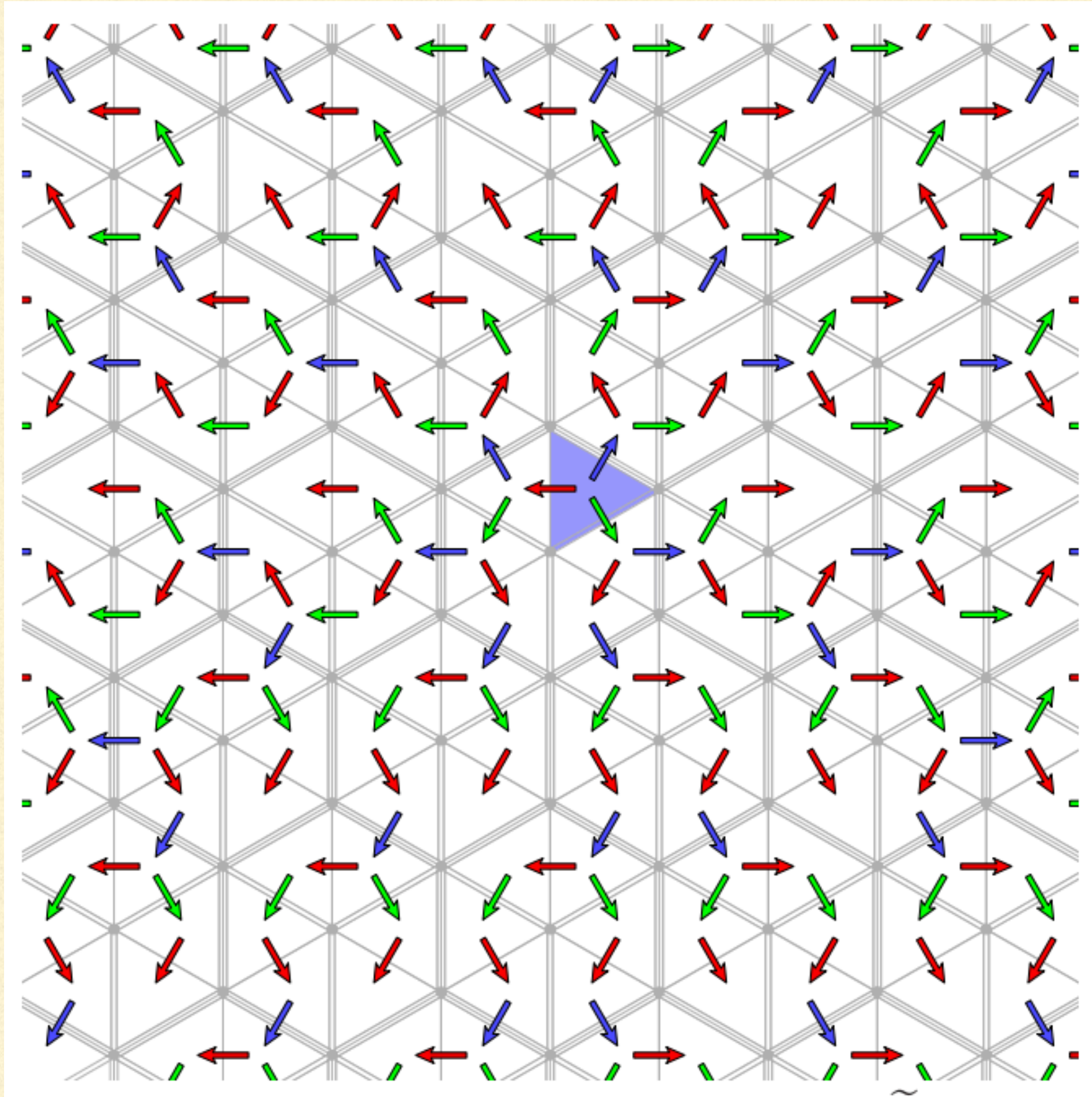
- Invariant slice of (3D) *canonical geometric realization* of $\tilde{A}_2$
- Dual graph of tiling is Cayley graph of $\tilde{A}_2$
- Fundamental domain (purple) is unique domain where 3 simple roots associated to s_R, s_G, s_B are all positive

AUTOMATA FOR COXETER GROUPS:



Lemma: (Casselman) Let C be the fundamental domain, $w \in W$, s_α be the reflection associated to a simple root α . Then s_α is a possible final letter of a shortest expression for w iff the dual hyperplane to α separates C from $w^{-1}C$ (i.e., iff α is negative on $w^{-1}C$).

AUTOMATA FOR COXETER GROUPS:

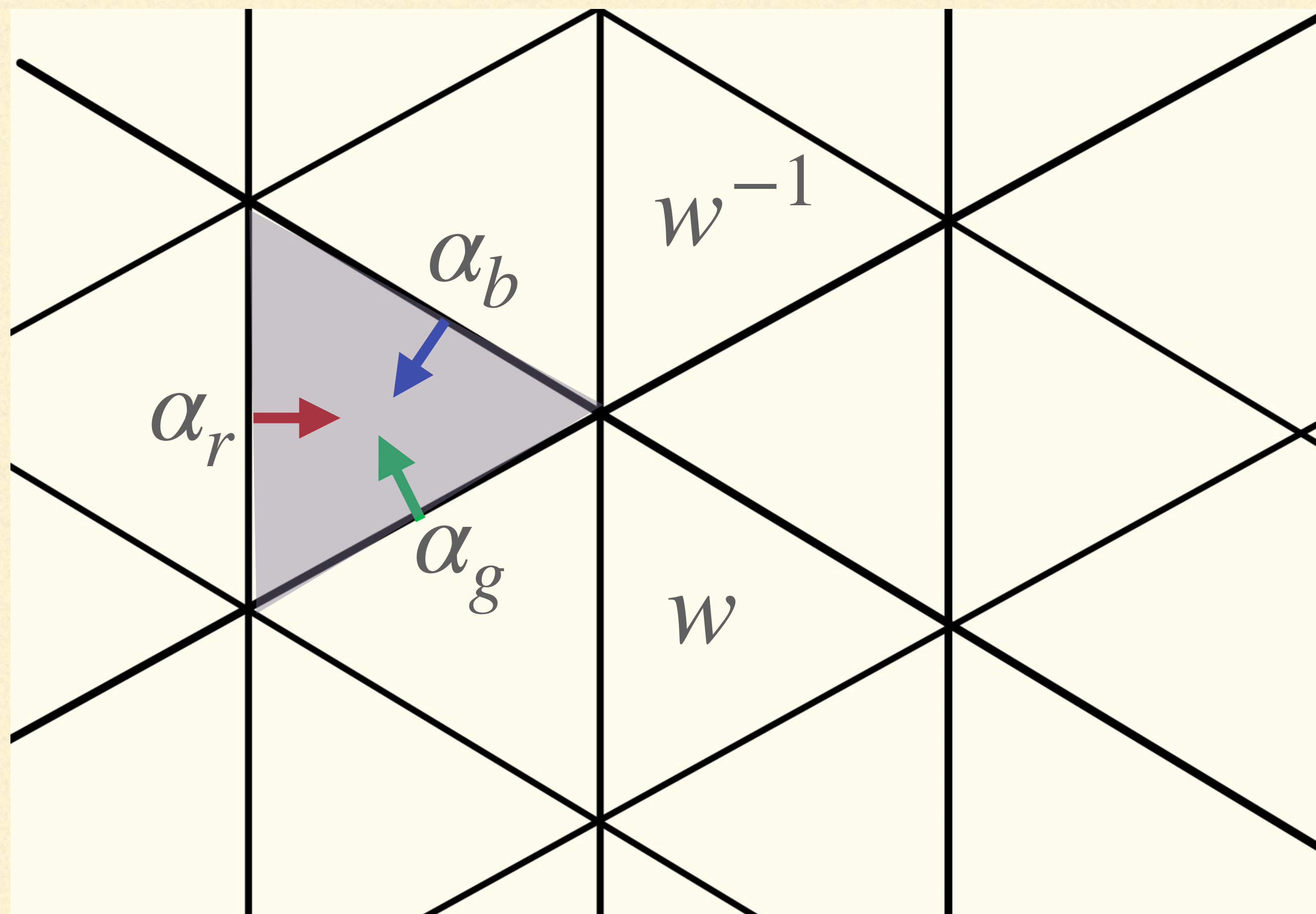


Lemma: (Casselman) Let C be the fundamental domain, $w \in W$, s_α be the reflection associated to a simple root α . Then s_α is a possible final letter of a shortest expression for w iff the dual hyperplane to α separates C from $w^{-1}C$ (i.e., iff α is negative on $w^{-1}C$).

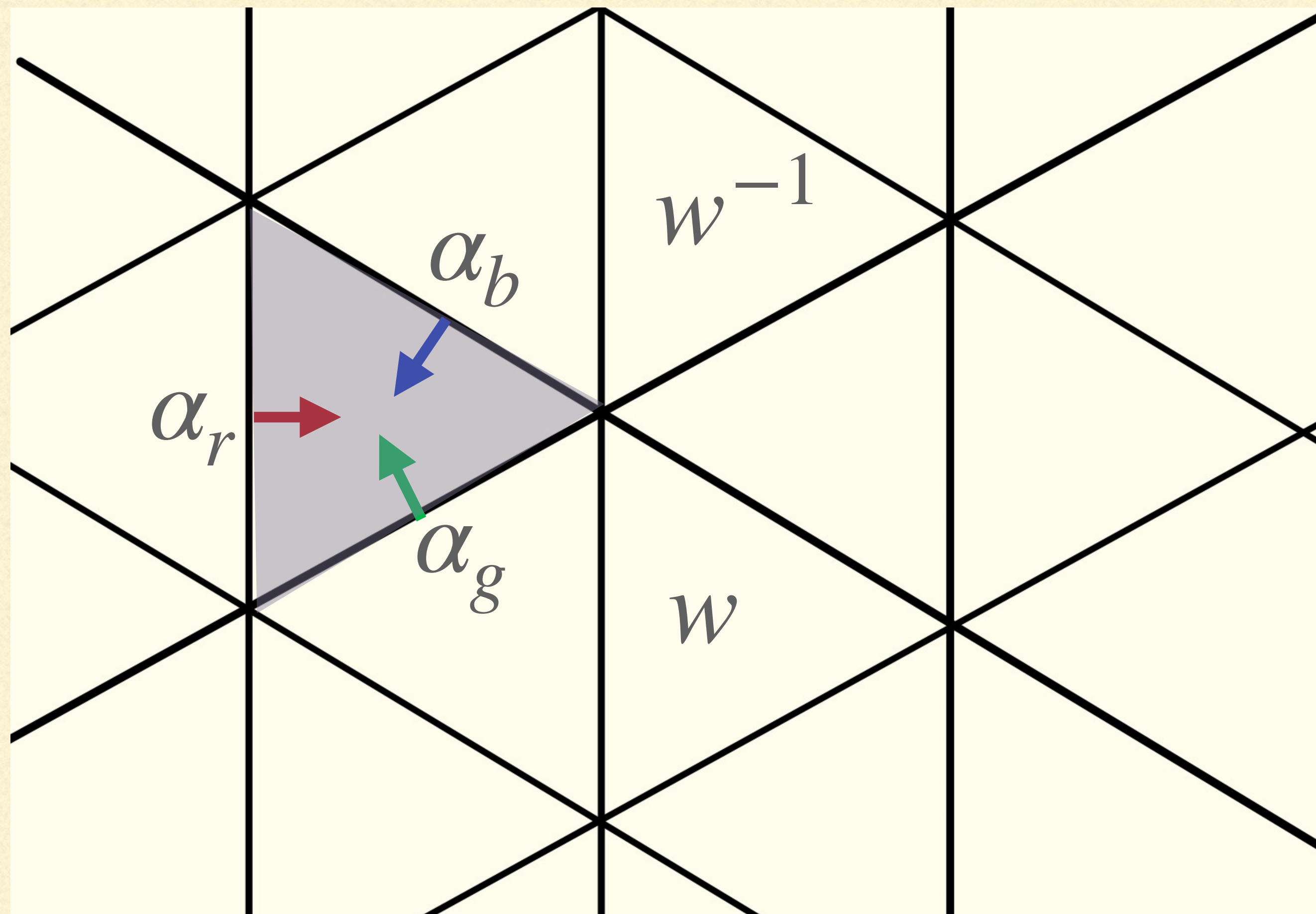
Corollary: (Algorithm for extracting the unique Inverse ShortLex expression for $w \in W$)

- Compute $\langle \alpha, w^{-1}C \rangle$ for all simple α
- Among α for which $\langle \alpha, w^{-1}C \rangle < 0$, pick lexicographically smallest

EXAMPLE: ($R < G < B$)



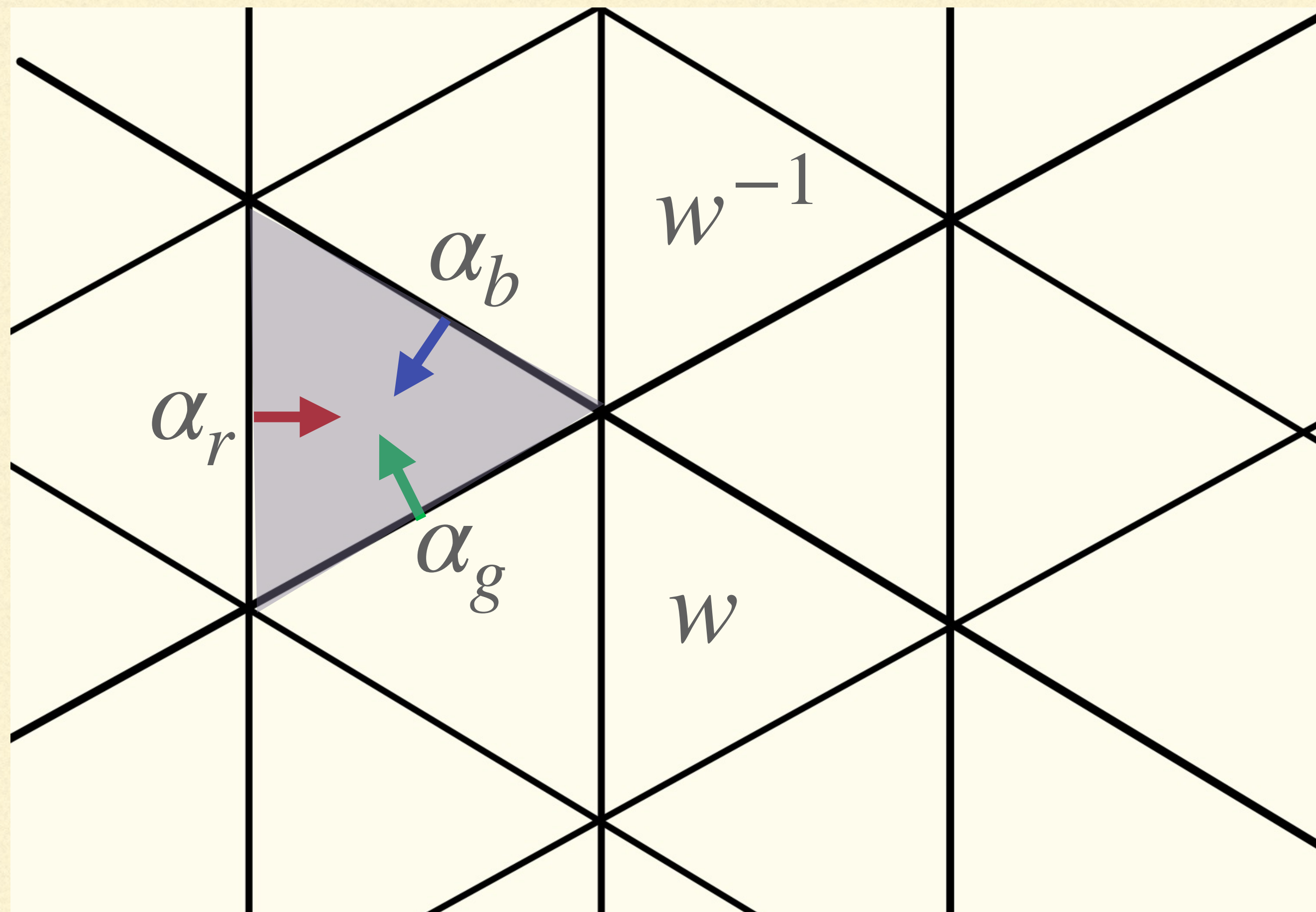
EXAMPLE: ($R < G < B$)



To write in Inverse Short Lex:

$$w = s_b s_g s_b s_g = ?$$

EXAMPLE: ($R < G < B$)

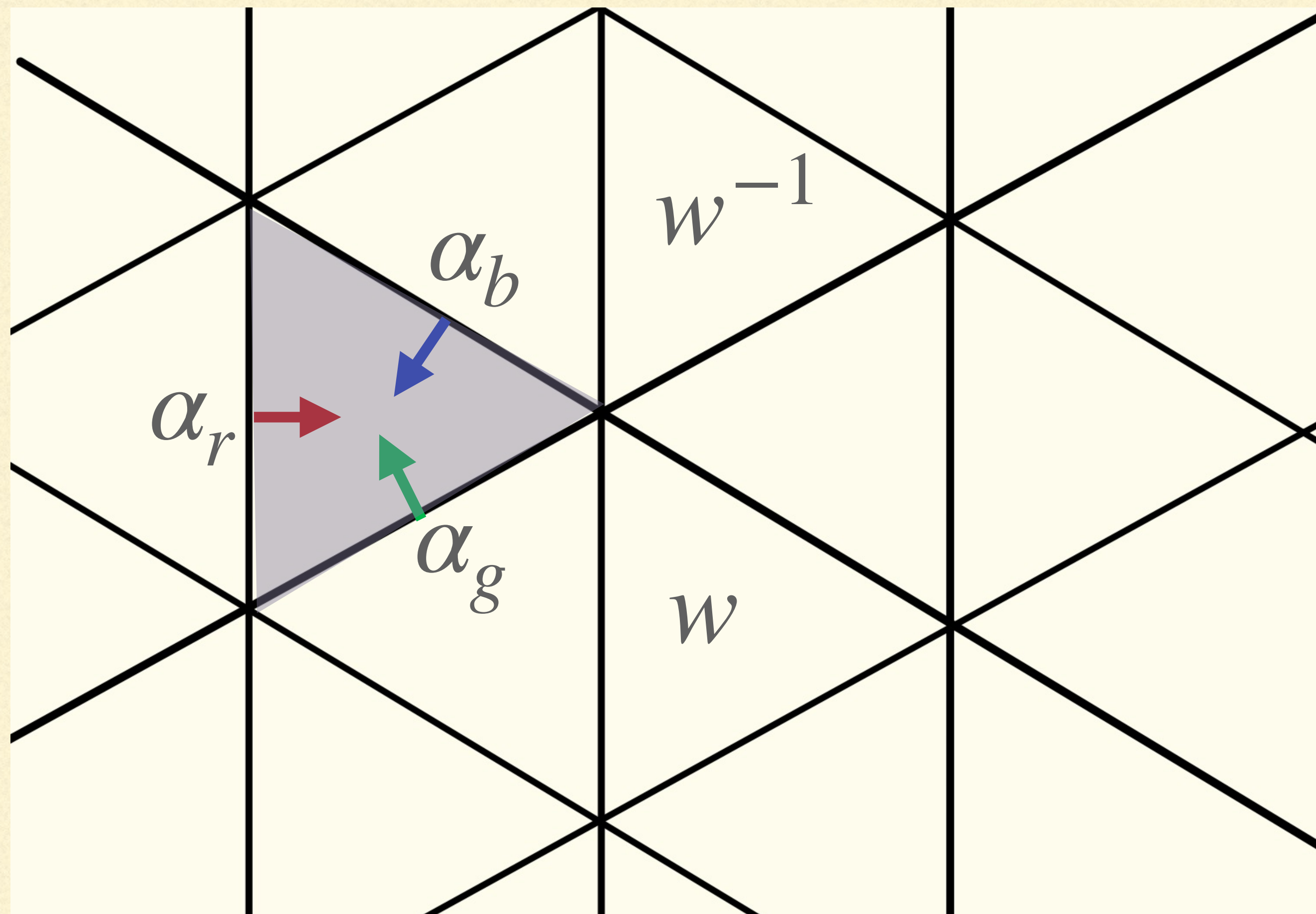


To write in Inverse Short Lex:

$$w = s_b s_g s_b s_g = ?$$

To write w in Inverse Short Lex:

EXAMPLE: ($R < G < B$)



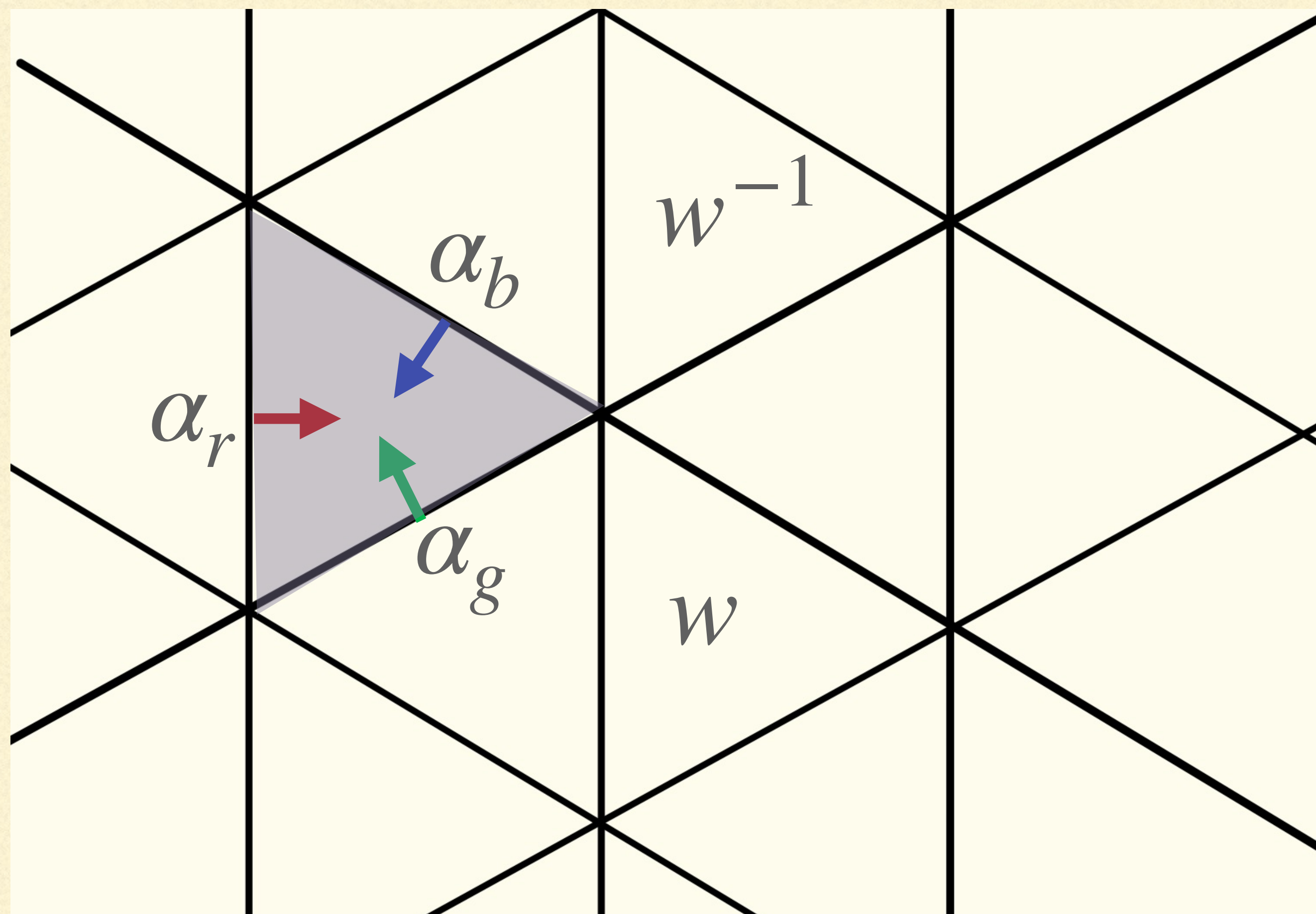
To write in Inverse Short Lex:

$$w = s_b s_g s_b s_g = ?$$

To write w in Inverse Short Lex:

Step 1: $w^{-1} = s_g s_b s_g s_b$

EXAMPLE: ($R < G < B$)



To write in Inverse Short Lex:

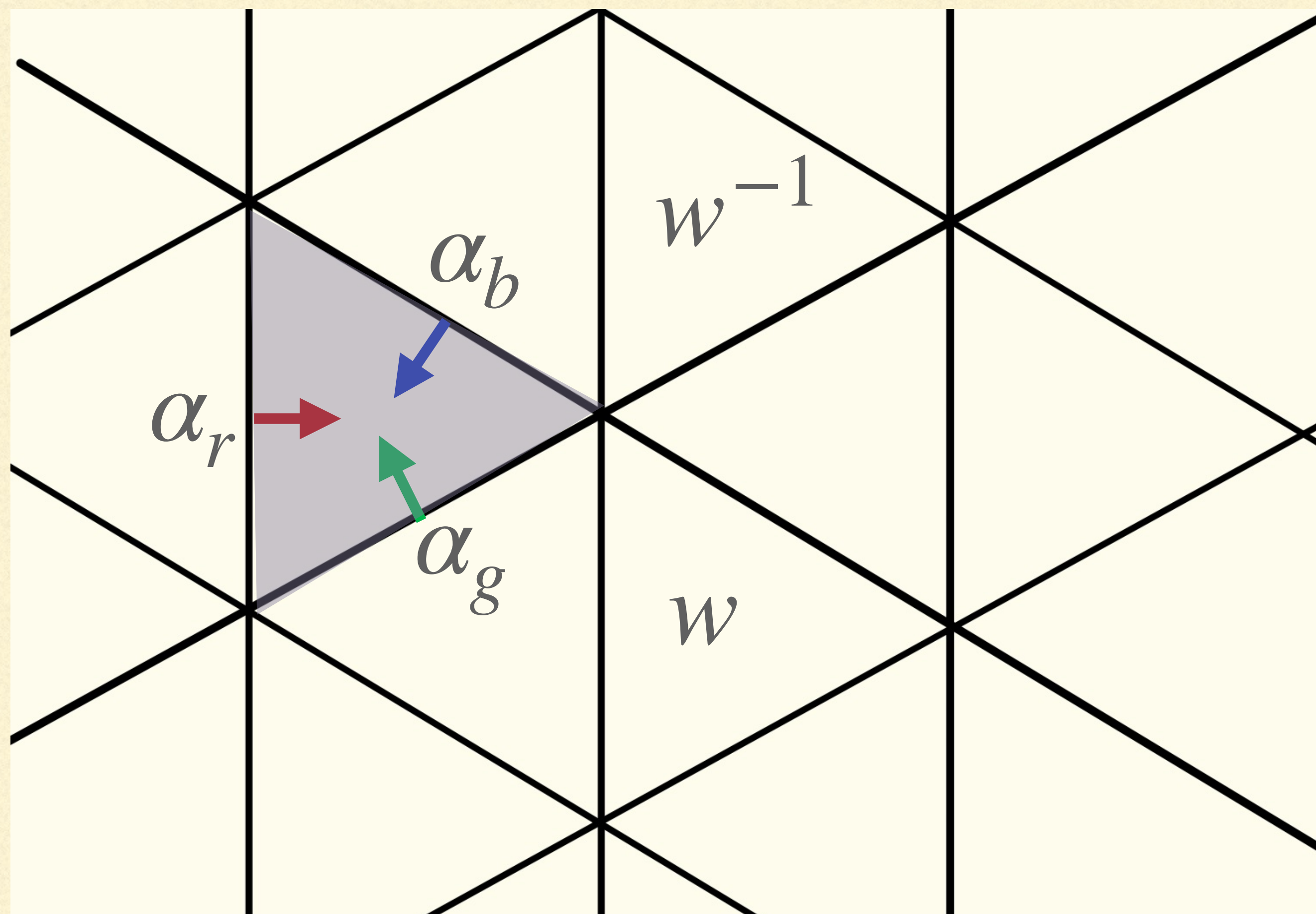
$$w = s_b s_g s_b s_g = ?$$

To write w in Inverse Short Lex:

Step 1: $w^{-1} = s_g s_b s_g s_b$

Step 2: Which roots are negative
at w^{-1} ?

EXAMPLE: ($R < G < B$)



To write in Inverse Short Lex:

$$w = s_b s_g s_b s_g = ?$$

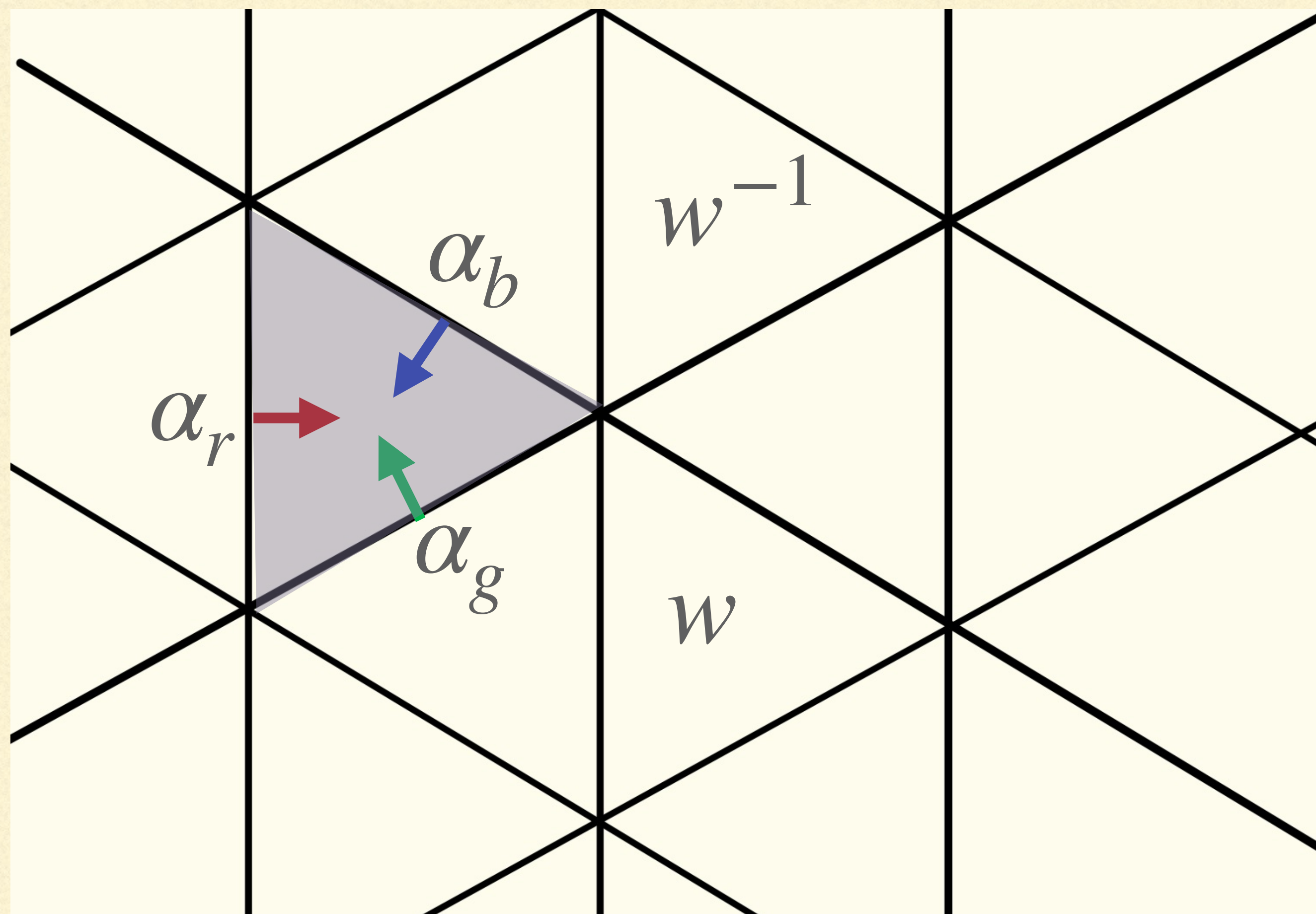
To write w in Inverse Short Lex:

Step 1: $w^{-1} = s_g s_b s_g s_b$

Step 2: Which roots are negative at w^{-1} ?

Step 3: Of these, which is least in lex order?

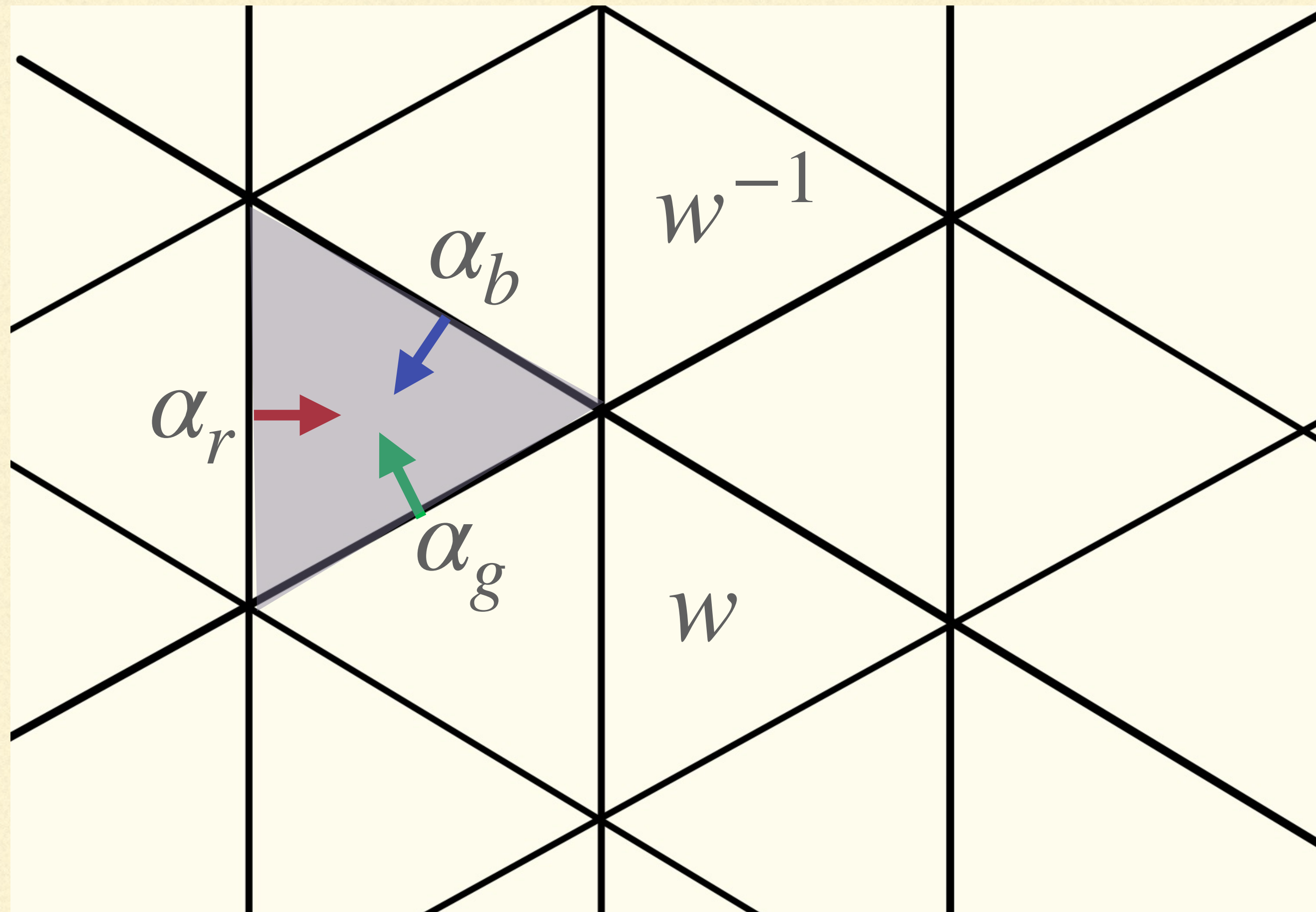
EXAMPLE: ($R < G < B$)



$$w = w's_b$$

So $w' = ws_b$

EXAMPLE: ($R < G < B$)



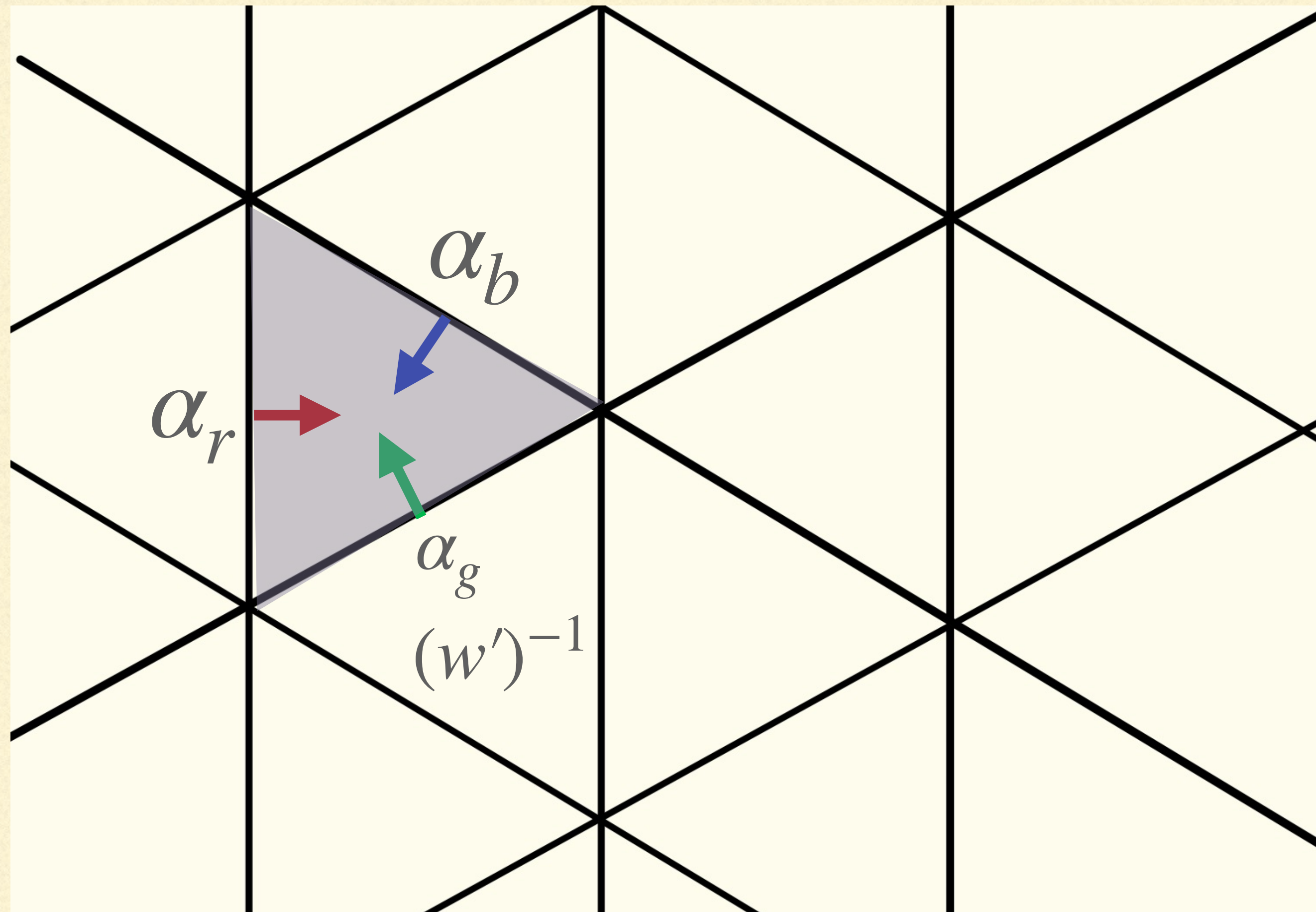
$$w = w's_b$$

$$\text{So } w' = ws_b$$

To write w' in Inverse Short Lex:

$$\text{Step 1: } (w')^{-1} = s_b s_g s_b s_g s_b$$

EXAMPLE: $(R < G < B)$



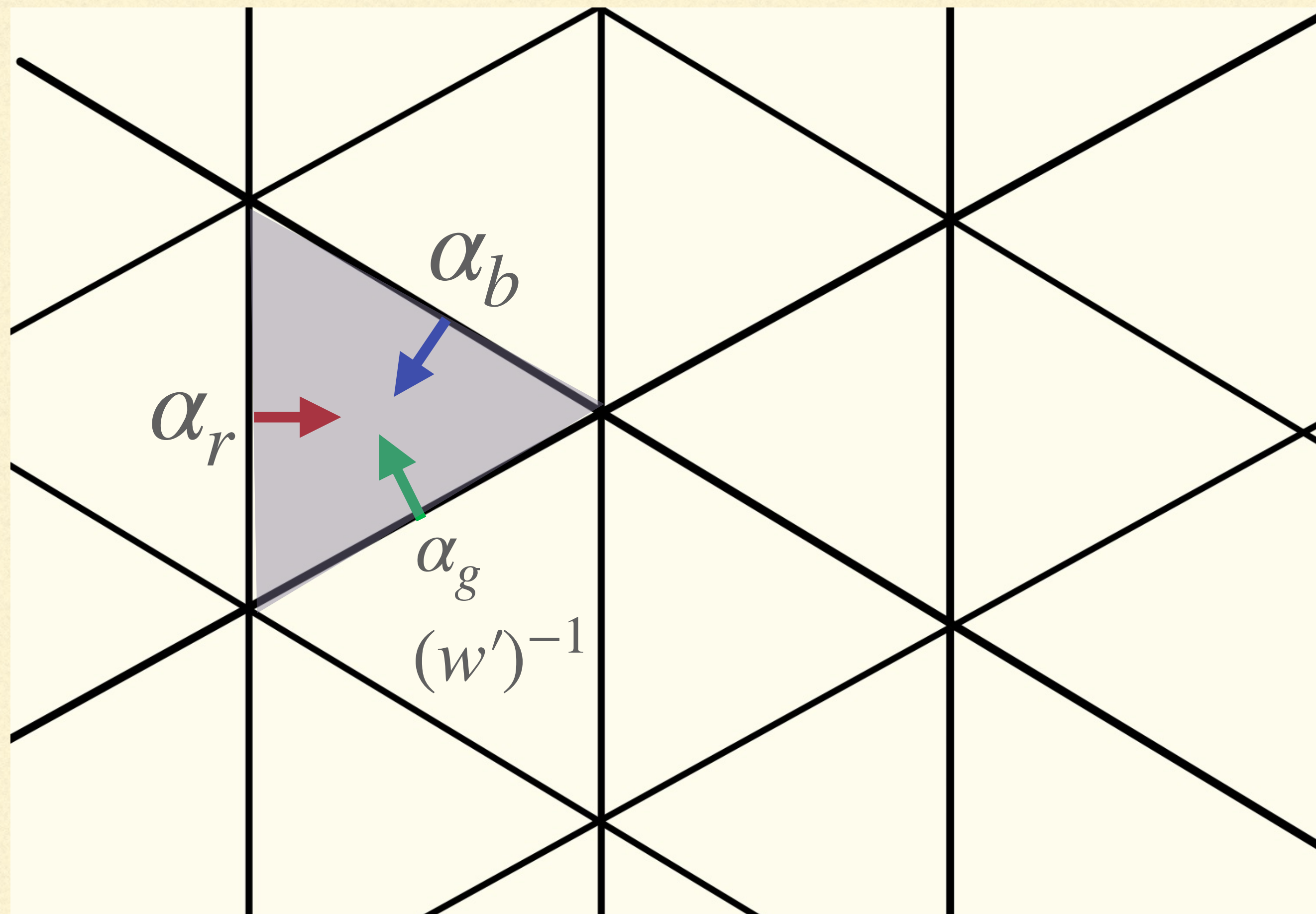
$$w = w's_b$$

$$\text{So } w' = ws_b$$

To write w' in Inverse Short Lex:

$$\text{Step 1: } (w')^{-1} = s_b s_g s_b s_g s_b$$

EXAMPLE: ($R < G < B$)



$$w = w's_b$$

$$\text{So } w' = ws_b$$

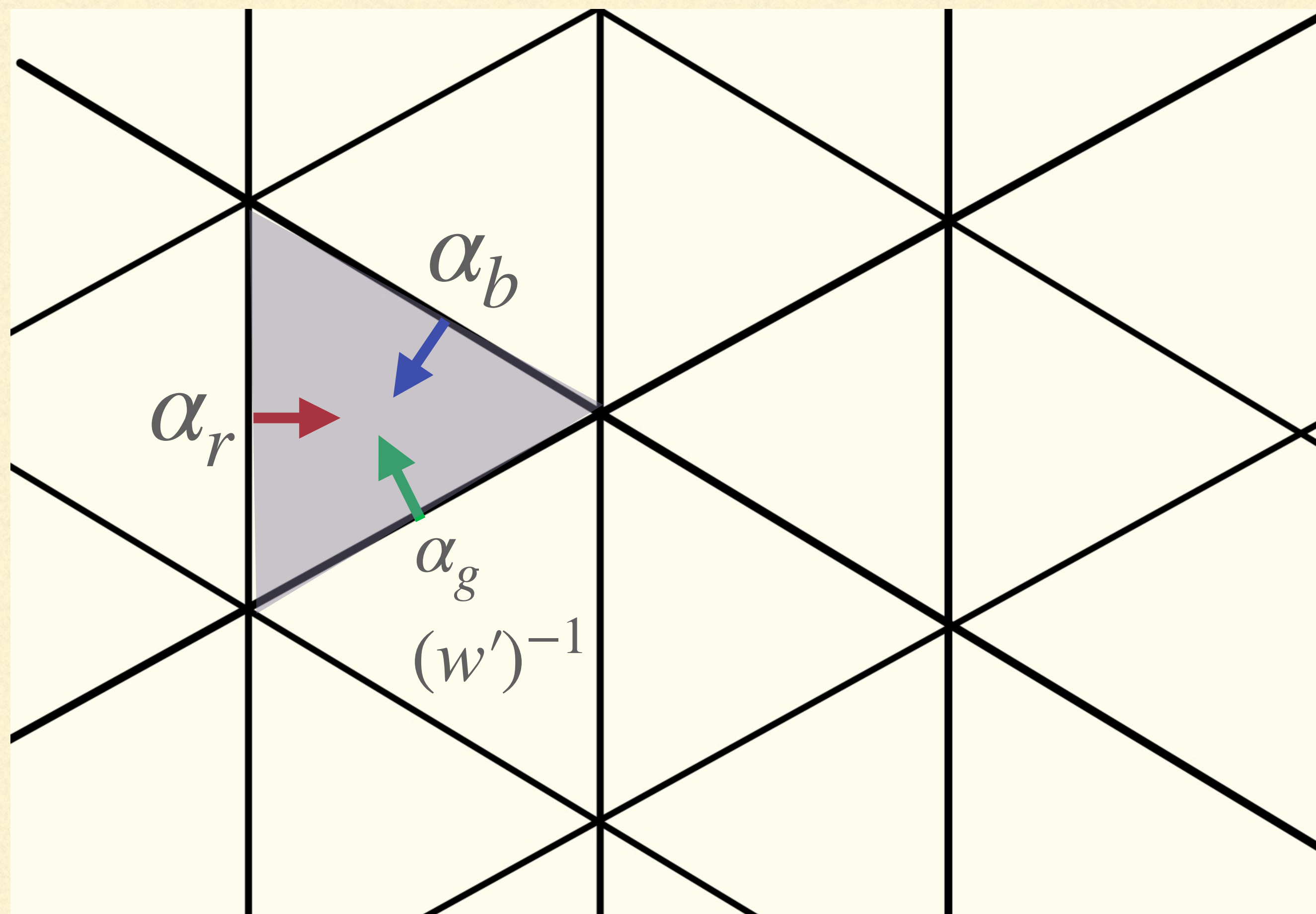
To write w' in Inverse Short Lex:

$$\text{Step 1: } (w')^{-1} = s_b s_g s_b s_g s_b$$

Step 2: Which roots are negative at w^{-1} ?

Step 3: Of these, which is least in lex order?

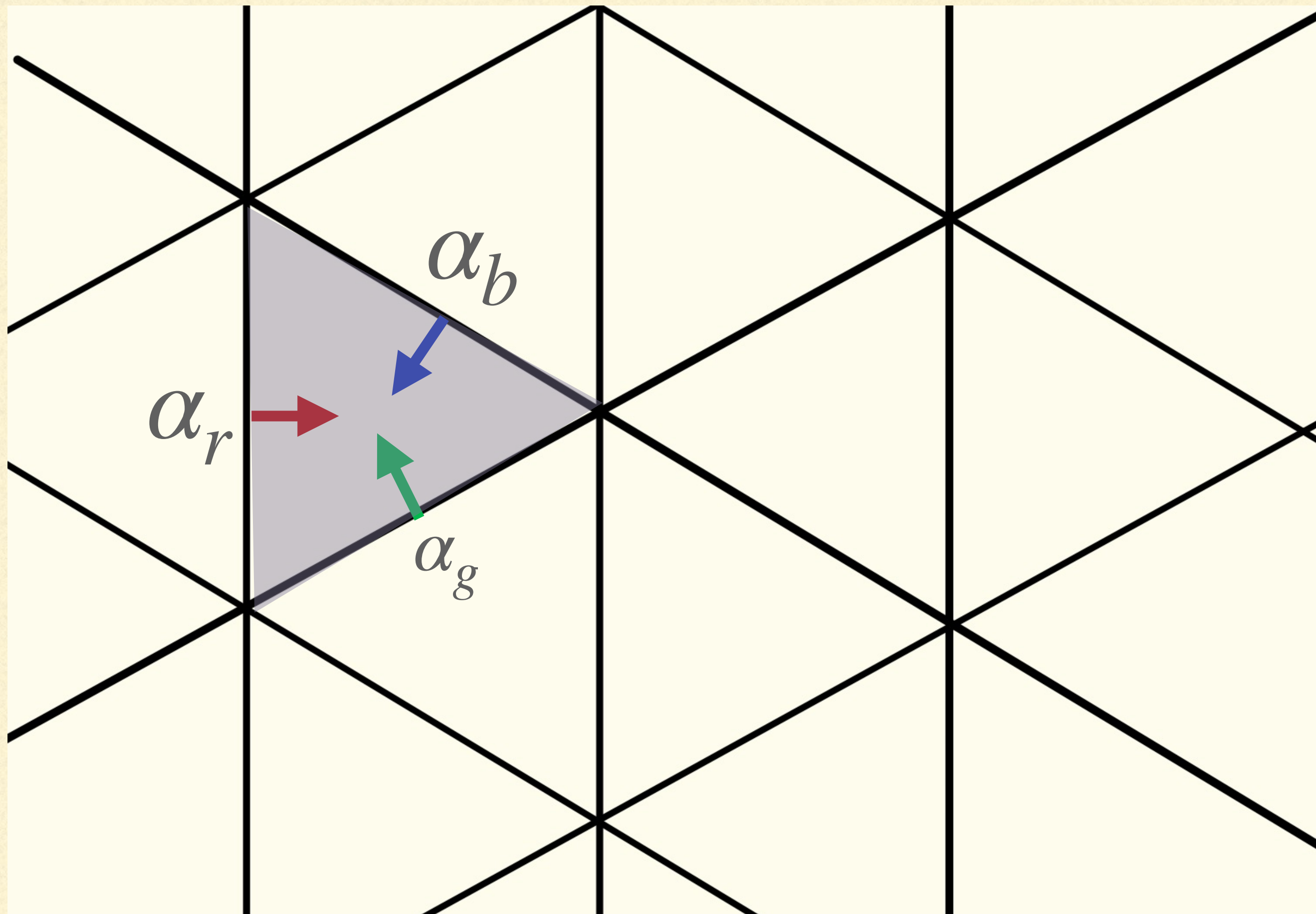
EXAMPLE: ($R < G < B$)



$$w = w'' s_g s_b$$

So $w'' = w s_b s_g$

EXAMPLE: ($R < G < B$)



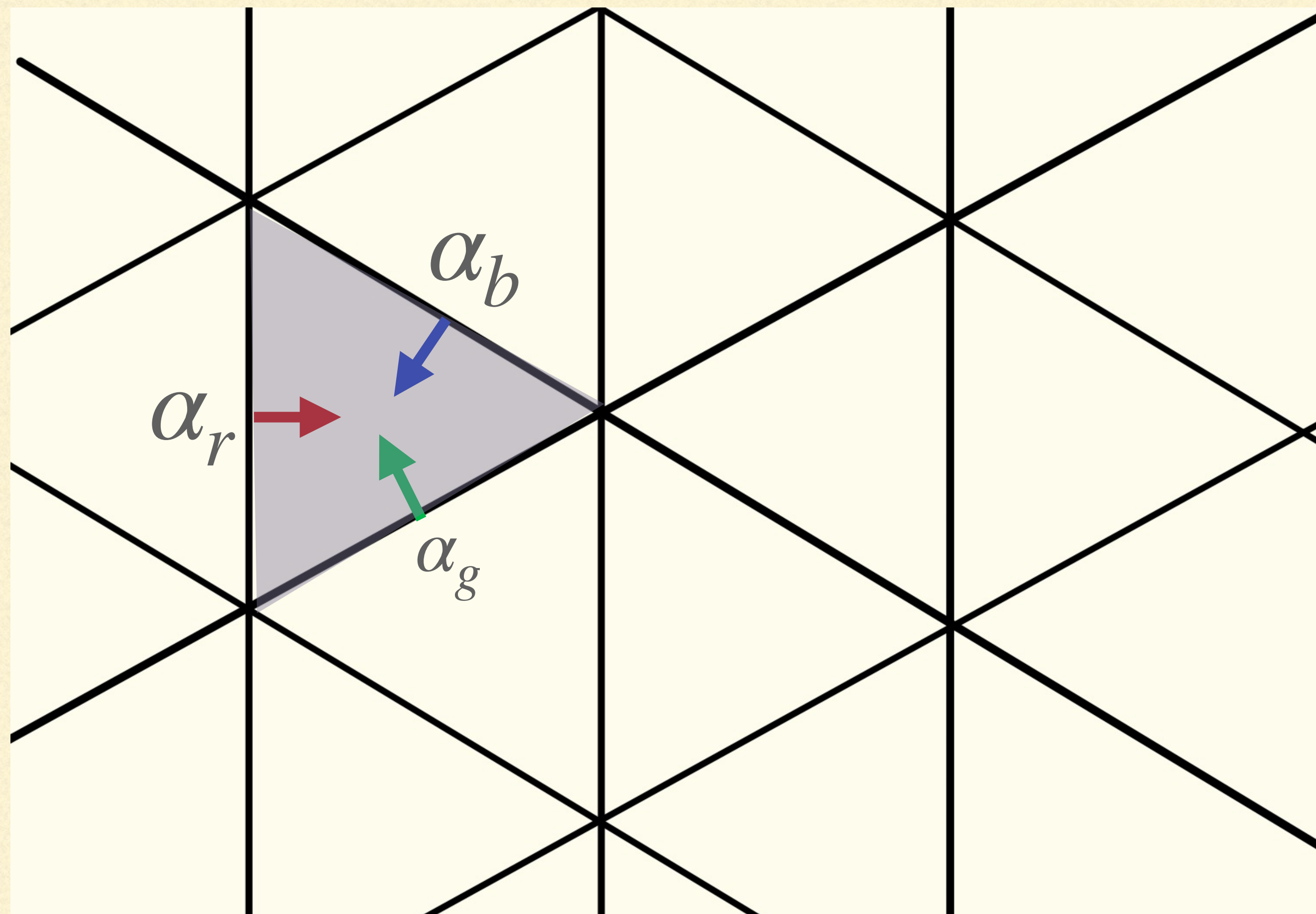
$$w = w'' s_g s_b$$

$$\text{So } w'' = w s_b s_g$$

To write w'' in Inverse Short Lex:

$$\text{Step 1: } (w'')^{-1} = s_g s_b s_g s_b s_g s_b$$

EXAMPLE: (R < G < B)



$$w = w'' s_g s_b$$

$$\text{So } w'' = w s_b s_g$$

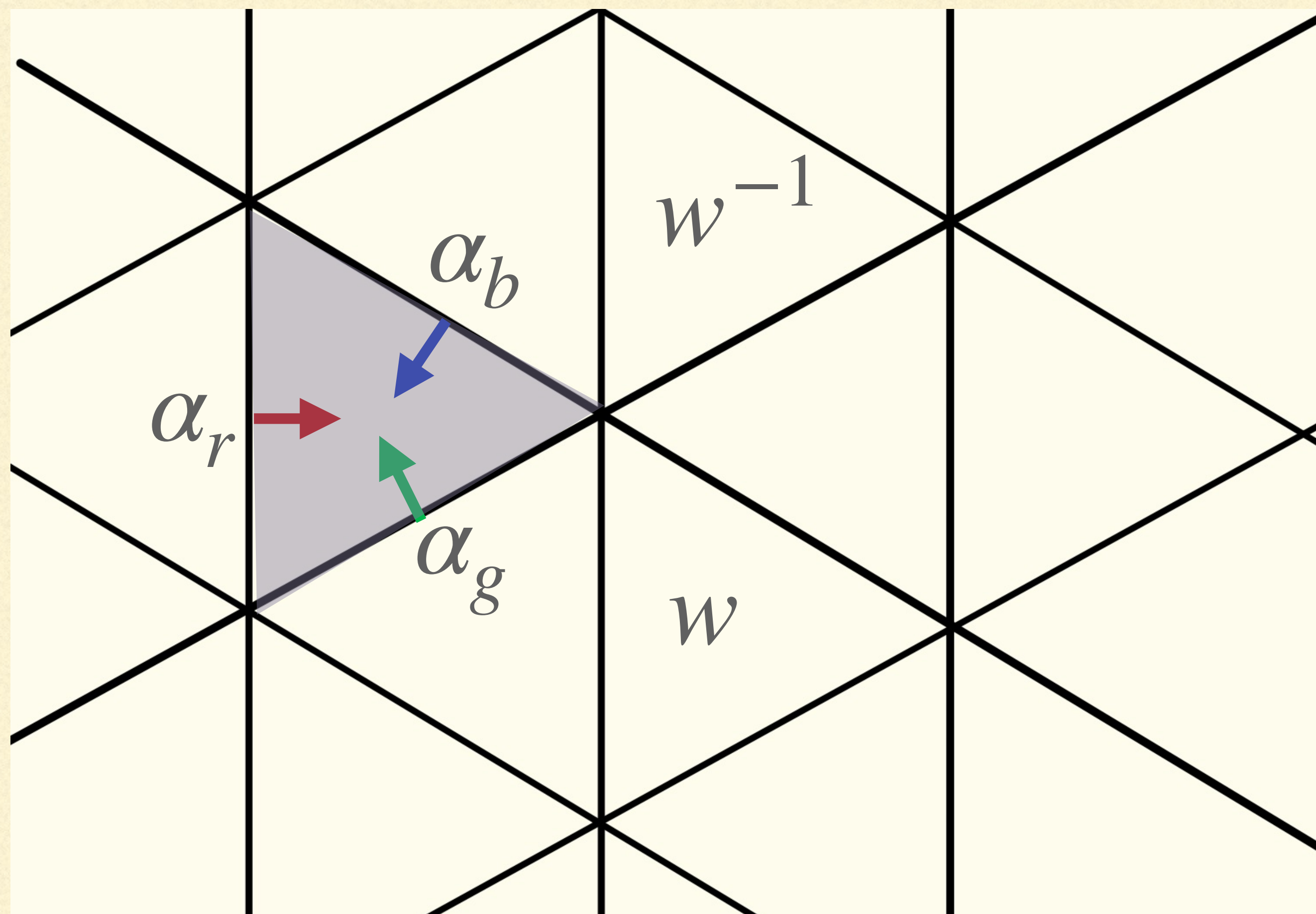
To write w'' in Inverse Short Lex:

$$\text{Step 1: } (w'')^{-1} = s_g s_b s_g s_b s_g s_b$$

Step 2: Which roots are negative at $(w'')^{-1}$?

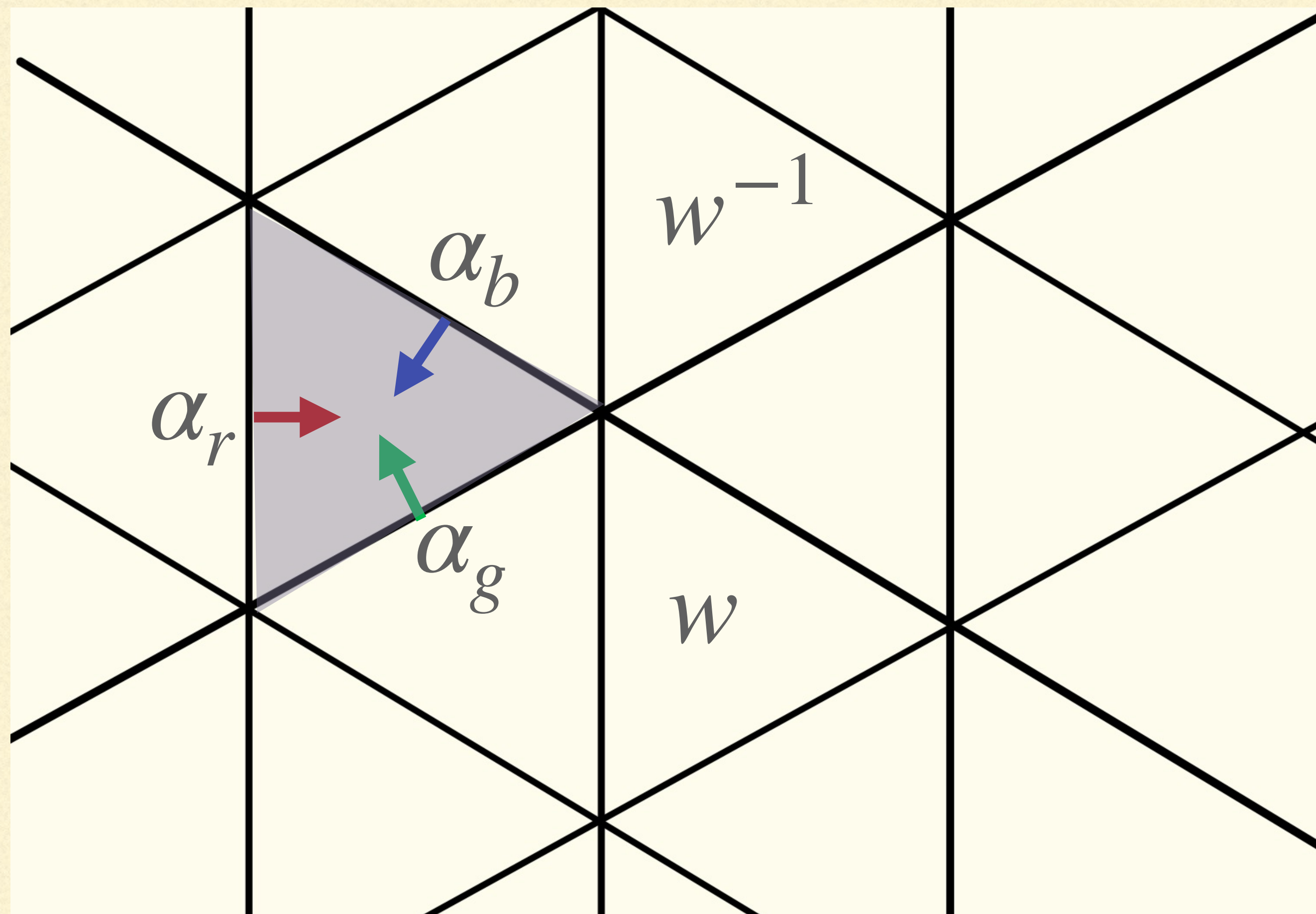
Step 3: Of these, which is least in lex order?

EXAMPLE: ($R < G < B$)



Inverse Short Lex expression:

EXAMPLE: ($R < G < B$)



Inverse Short Lex expression:

$$w = s_g s_b$$

TWO TYPES OF TRAINING DATA:

Type I:

End-binary labeled (state after final input token):

$\{1, 2, 1, 3, 2, 1, 2, 1, 2, 1, 3, 1, 2, 1\} \longrightarrow \{0\}$ Trivial

$\{1, 2, 1, 3, 2, 1, 2, 1, 2, 1\} \longrightarrow \{1\}$ Nontrivial

TWO TYPES OF TRAINING DATA:

Type 2:

Sequentially labeled (after each token):

$$S_3 = \{e, s_1, s_2, s_1s_2, s_2s_1, s_1s_2s_1\} = \{e, 1, 2, 12, 21, 121\}$$

Complete S_3 training data:

Example input string: (2 , 2 , 1 , 2 , 1 , 2 , 2)

Example output string: (2 , e , 1 , 12 , 121 , 21 , 121)

BIGGEST CONTRIBUTORS TO SUCCESSFUL LEARNING:

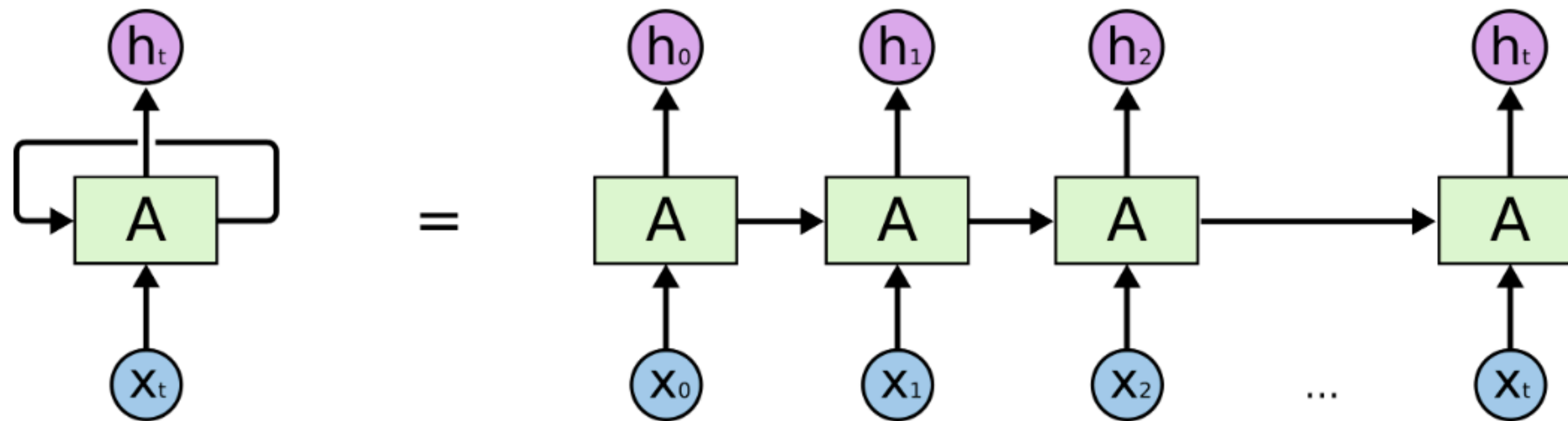
- Quality of training data set

Type 2 (sequentially-labeled) better than Type 1 (end-labeled)

- Difficulty of the problem

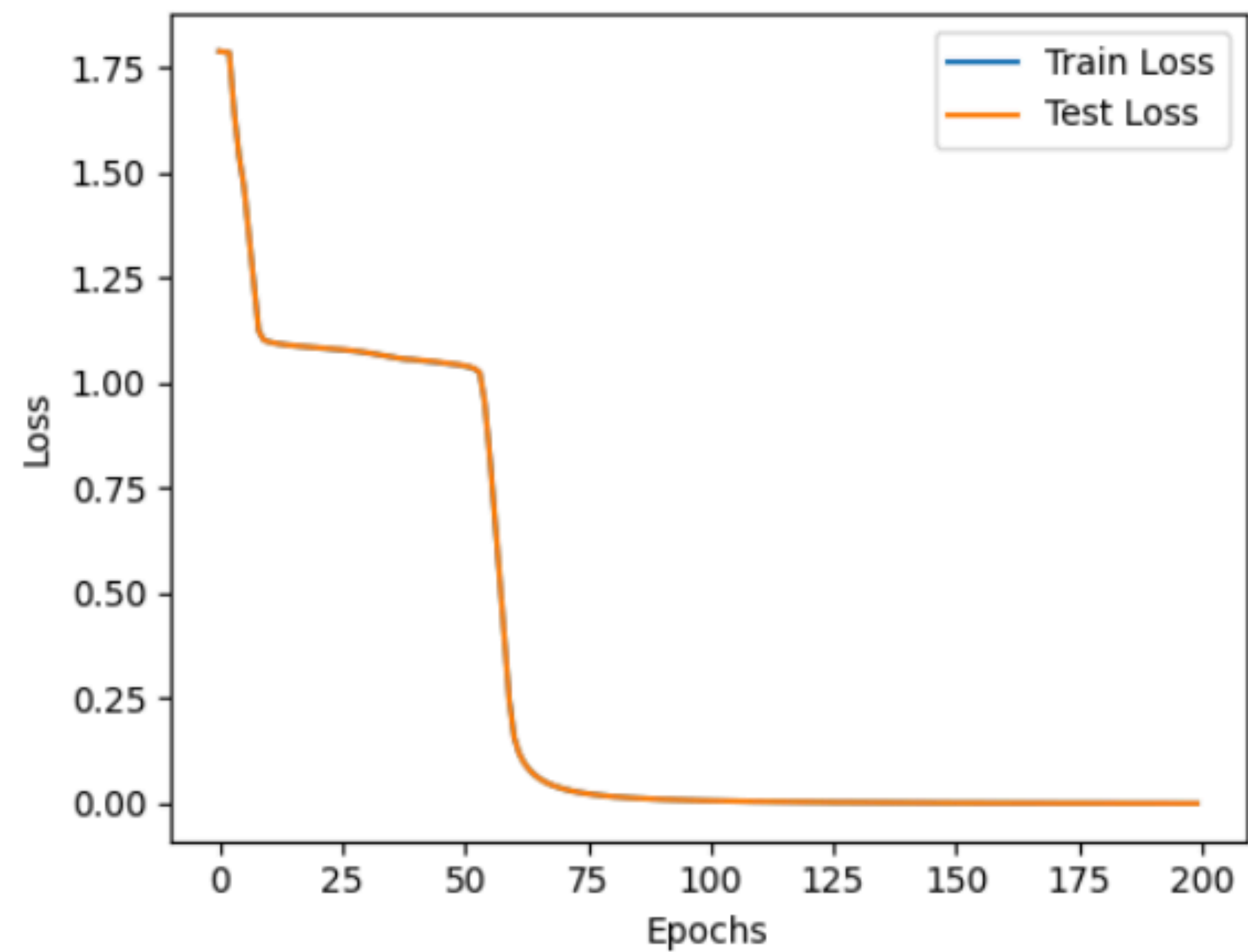
The word problem is much easier in S_3 than $\tilde{A}_2$

RECURRENT NEURAL NETWORK:

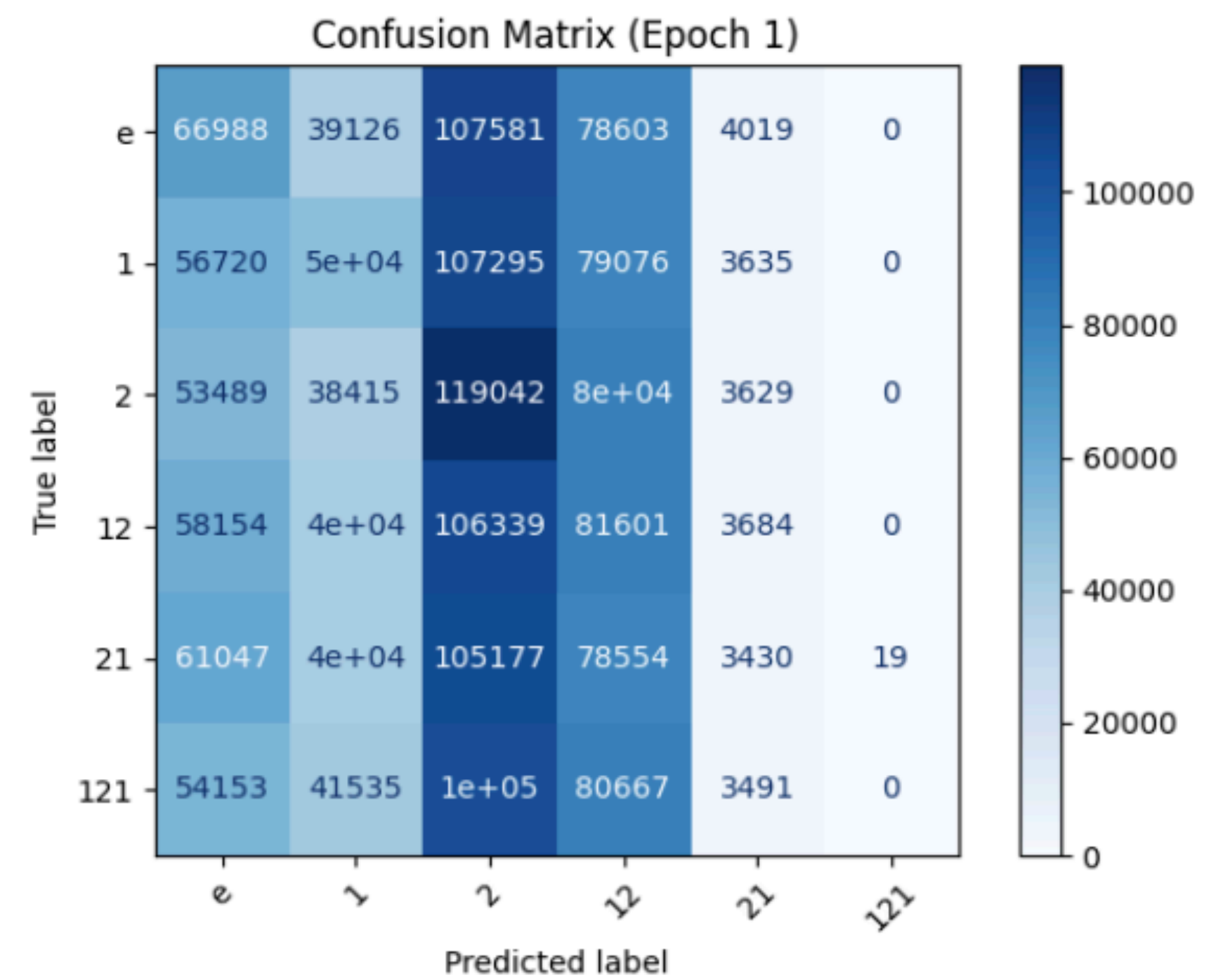
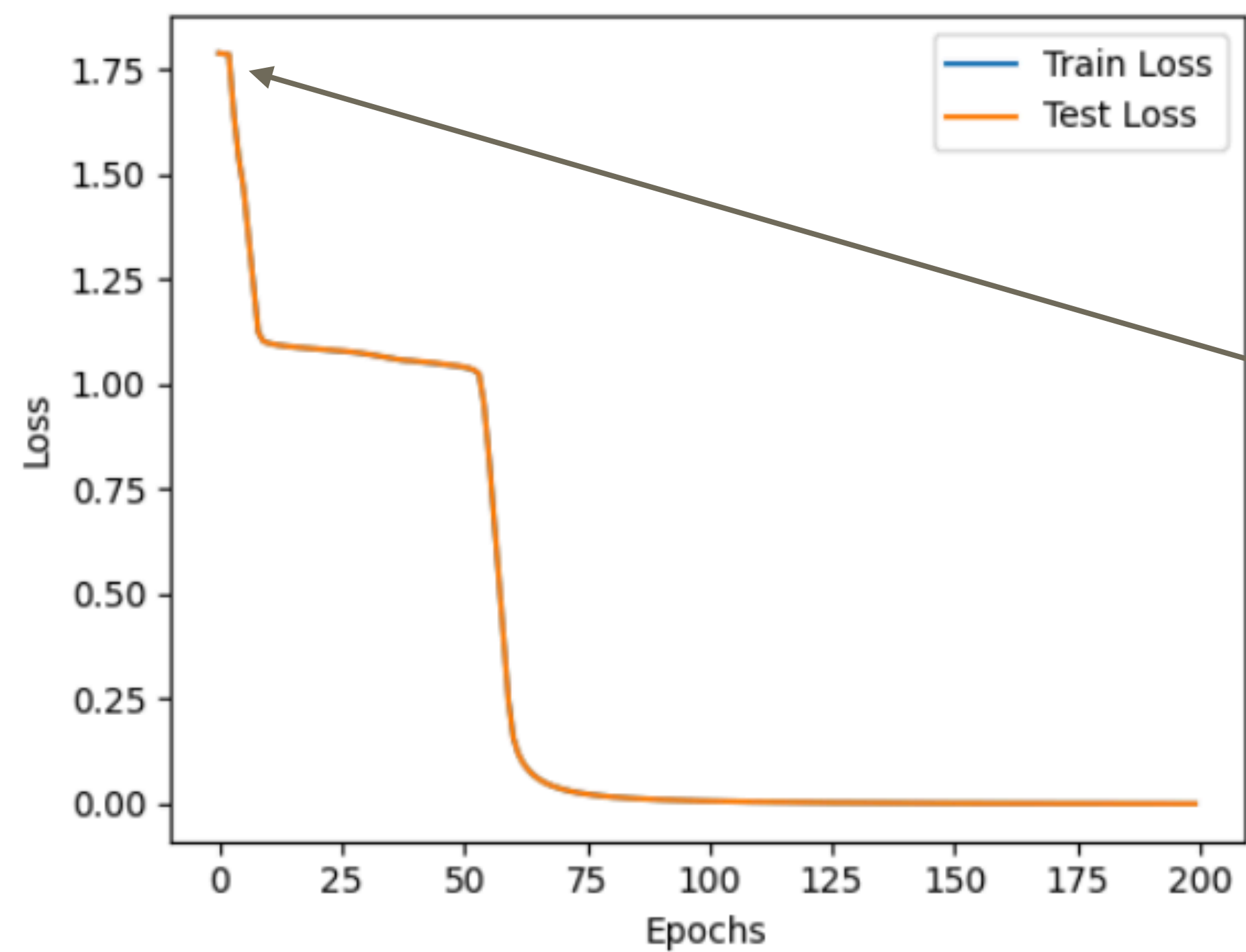


An unrolled recurrent neural network.

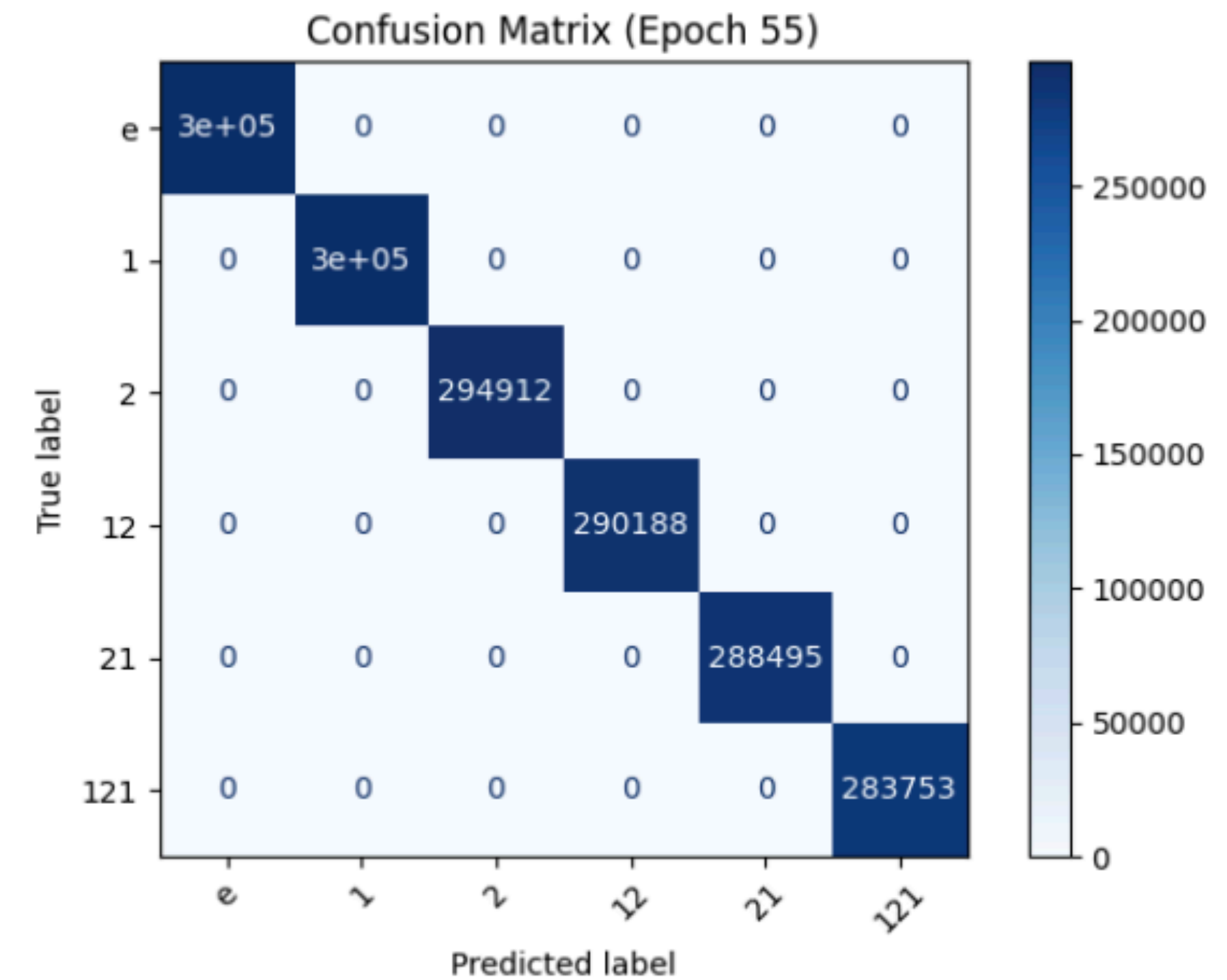
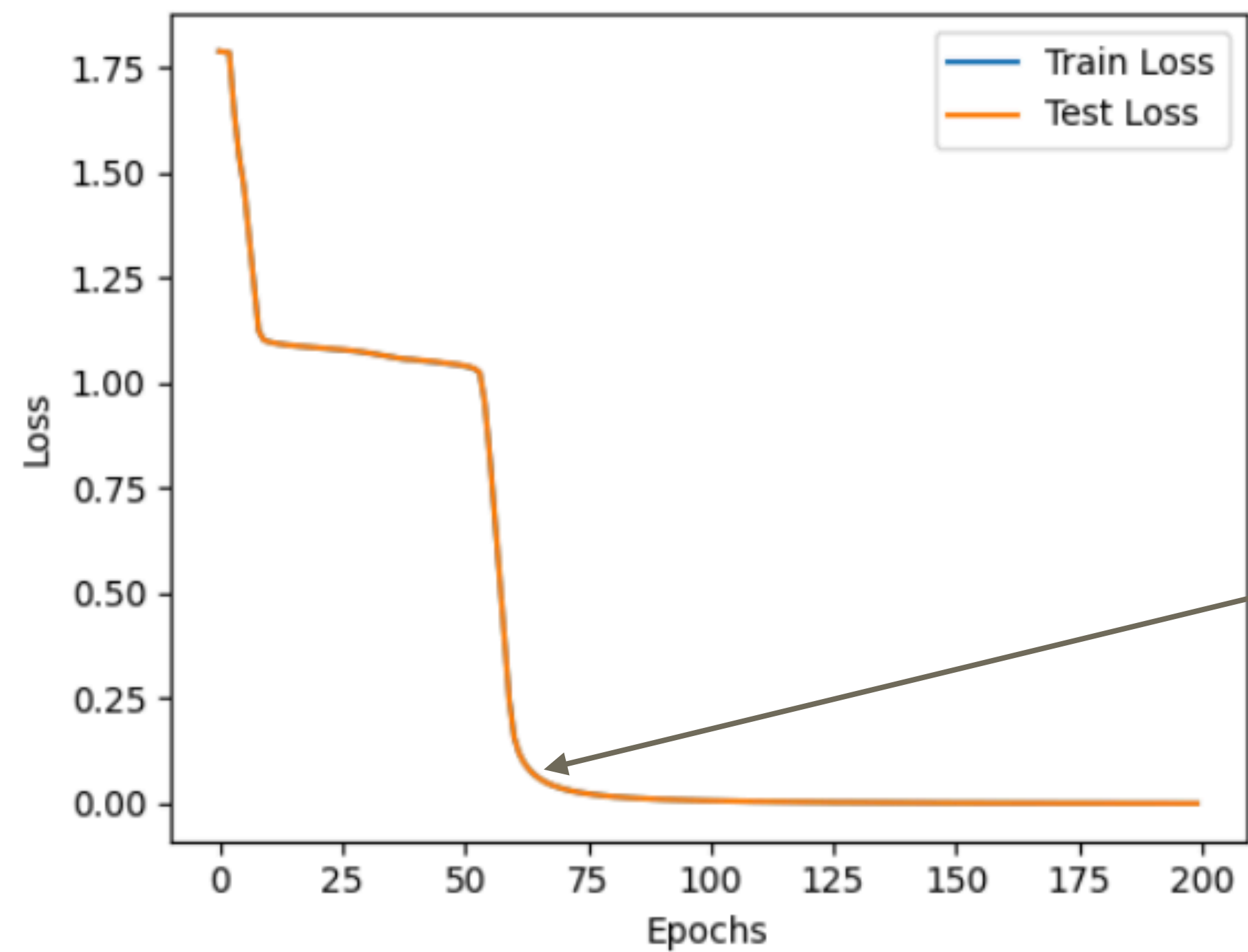
RECURRENT NEURAL NETWORK:



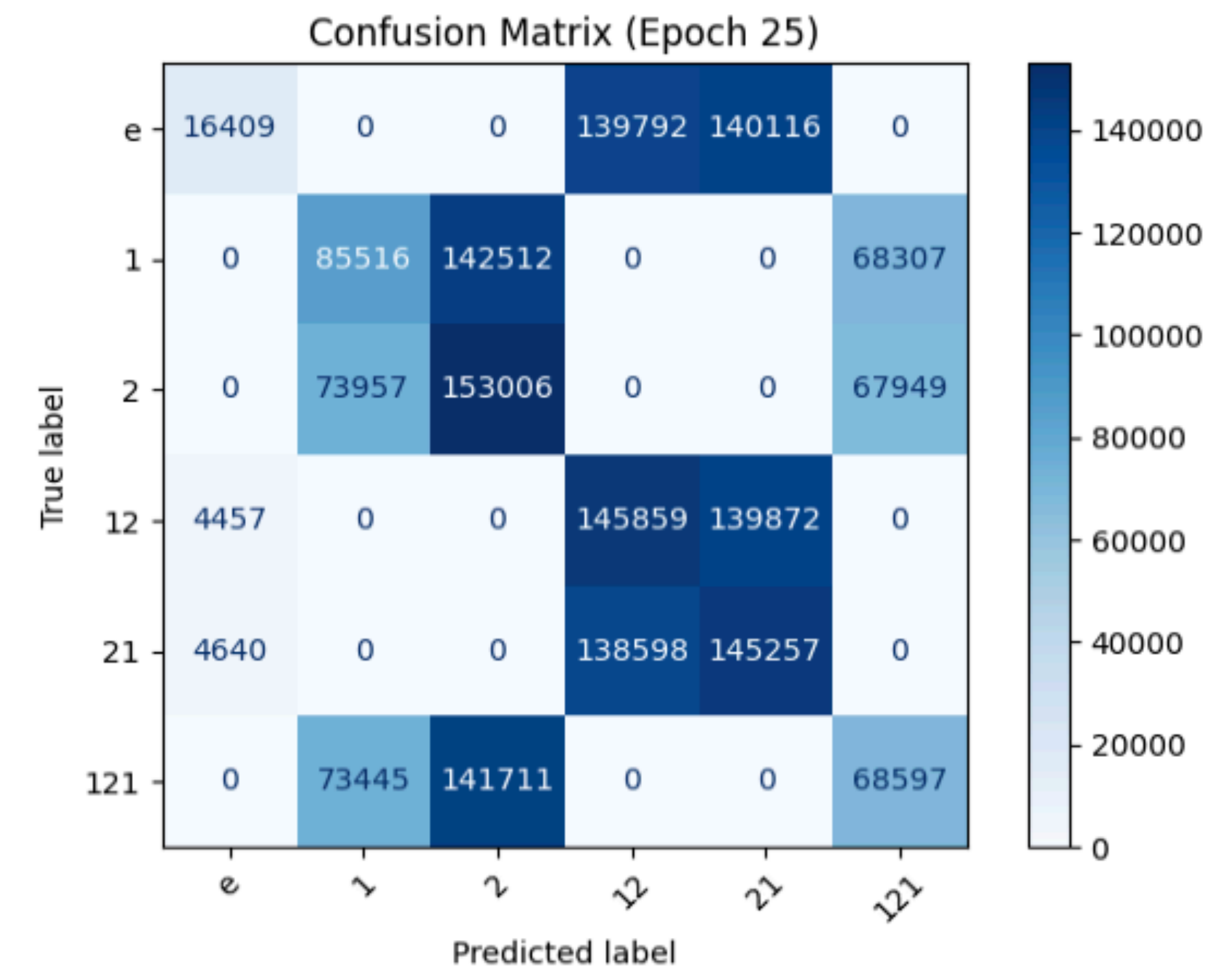
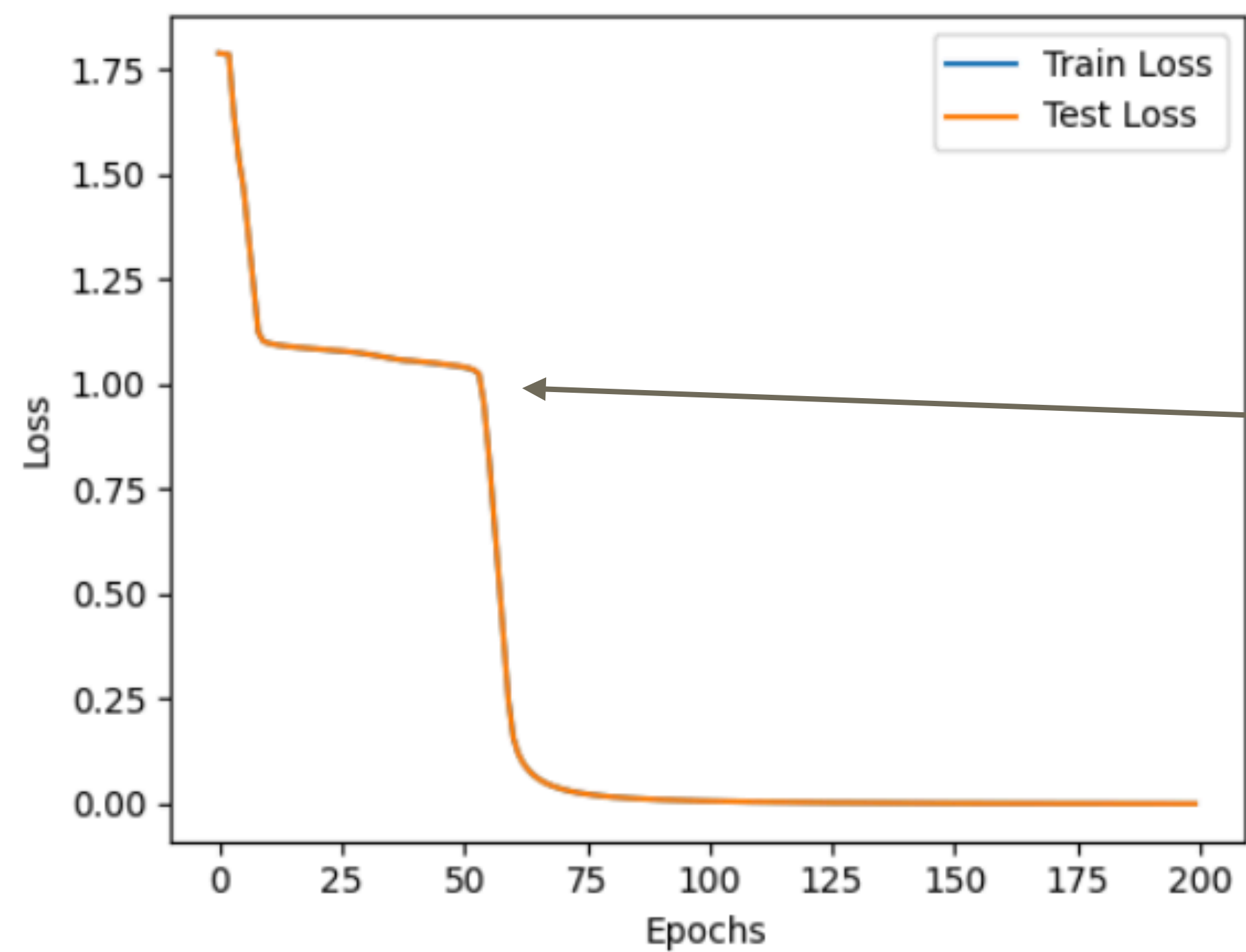
RECURRENT NEURAL NETWORK:



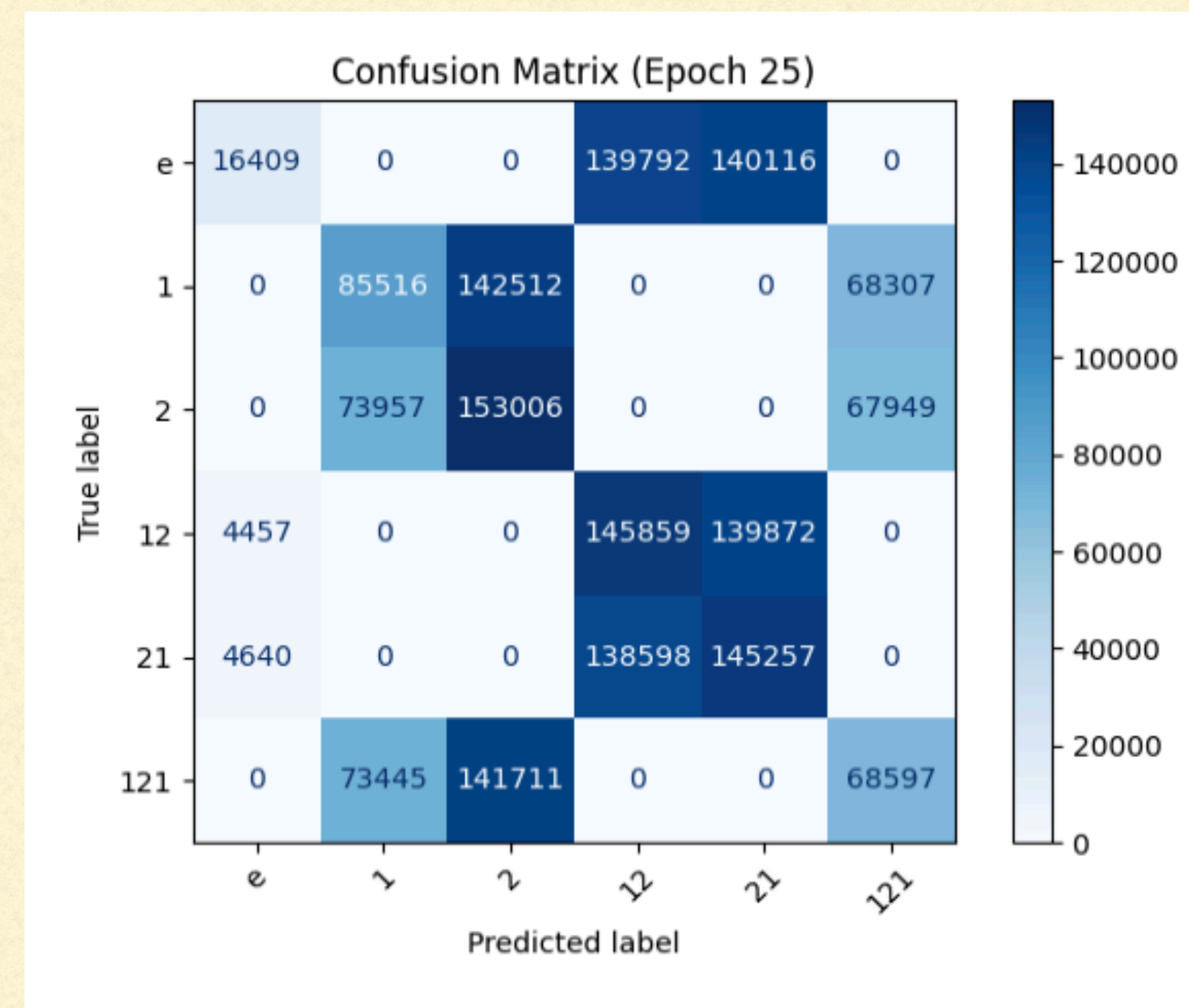
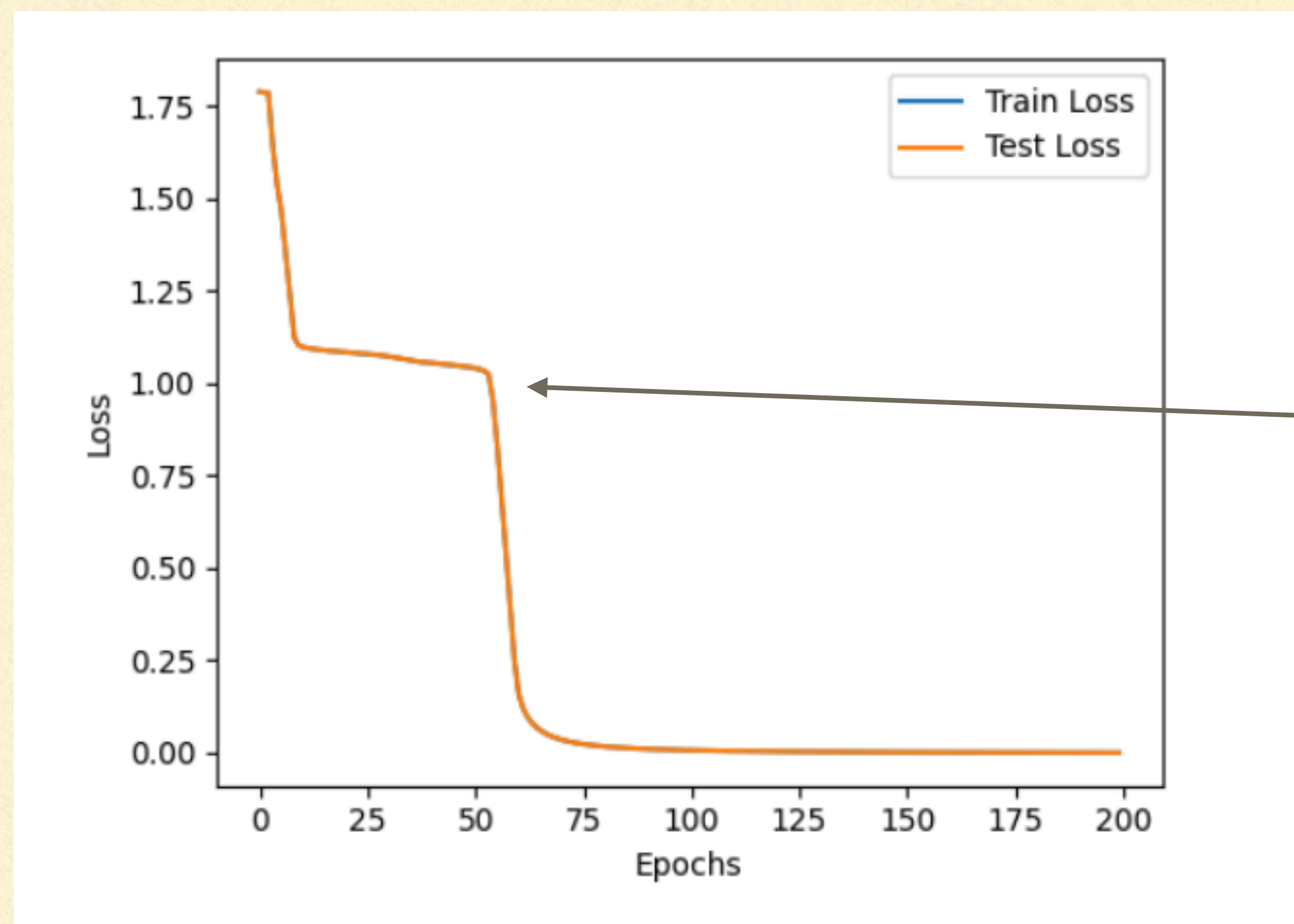
RECURRENT NEURAL NETWORK:



RECURRENT NEURAL NETWORK:

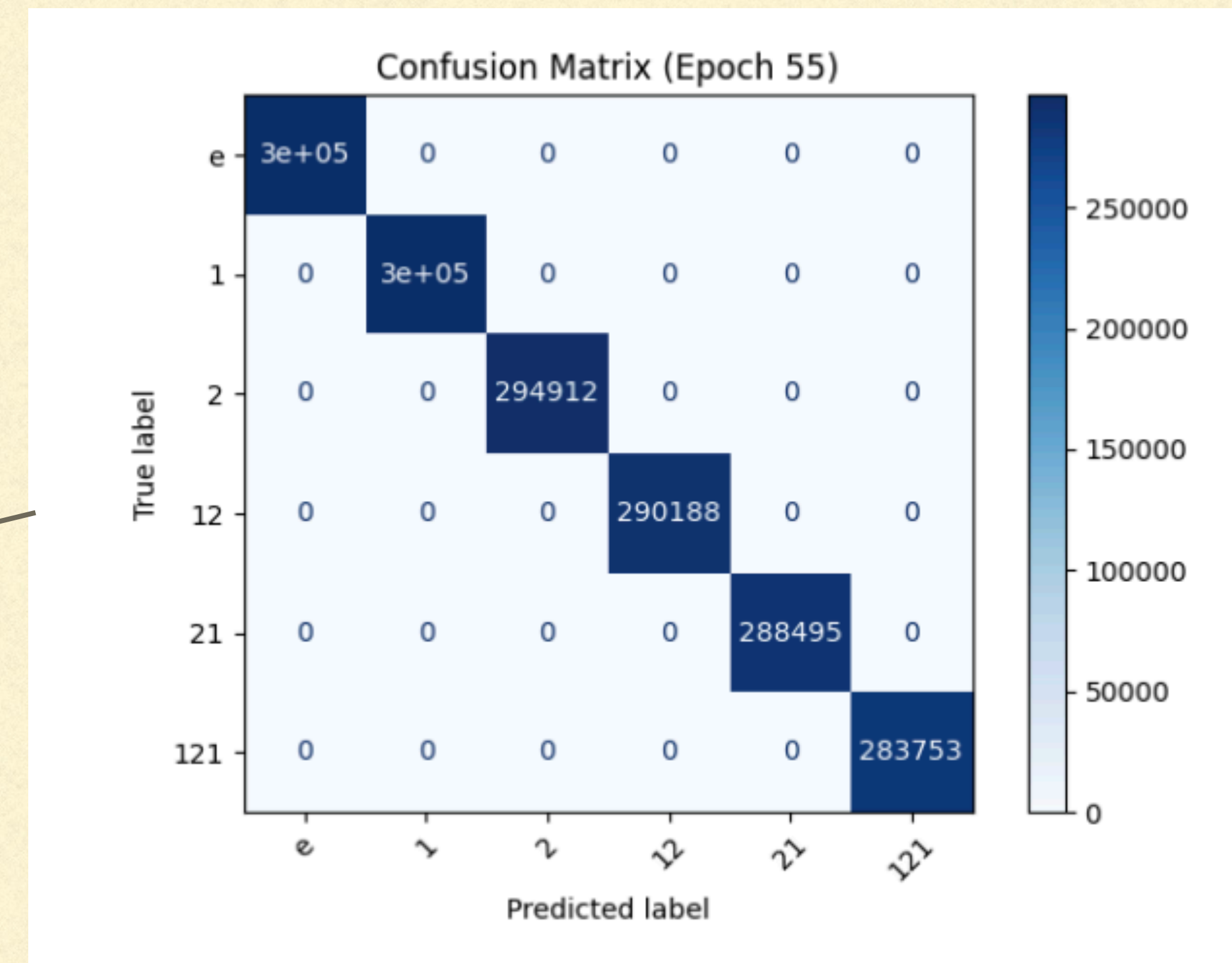
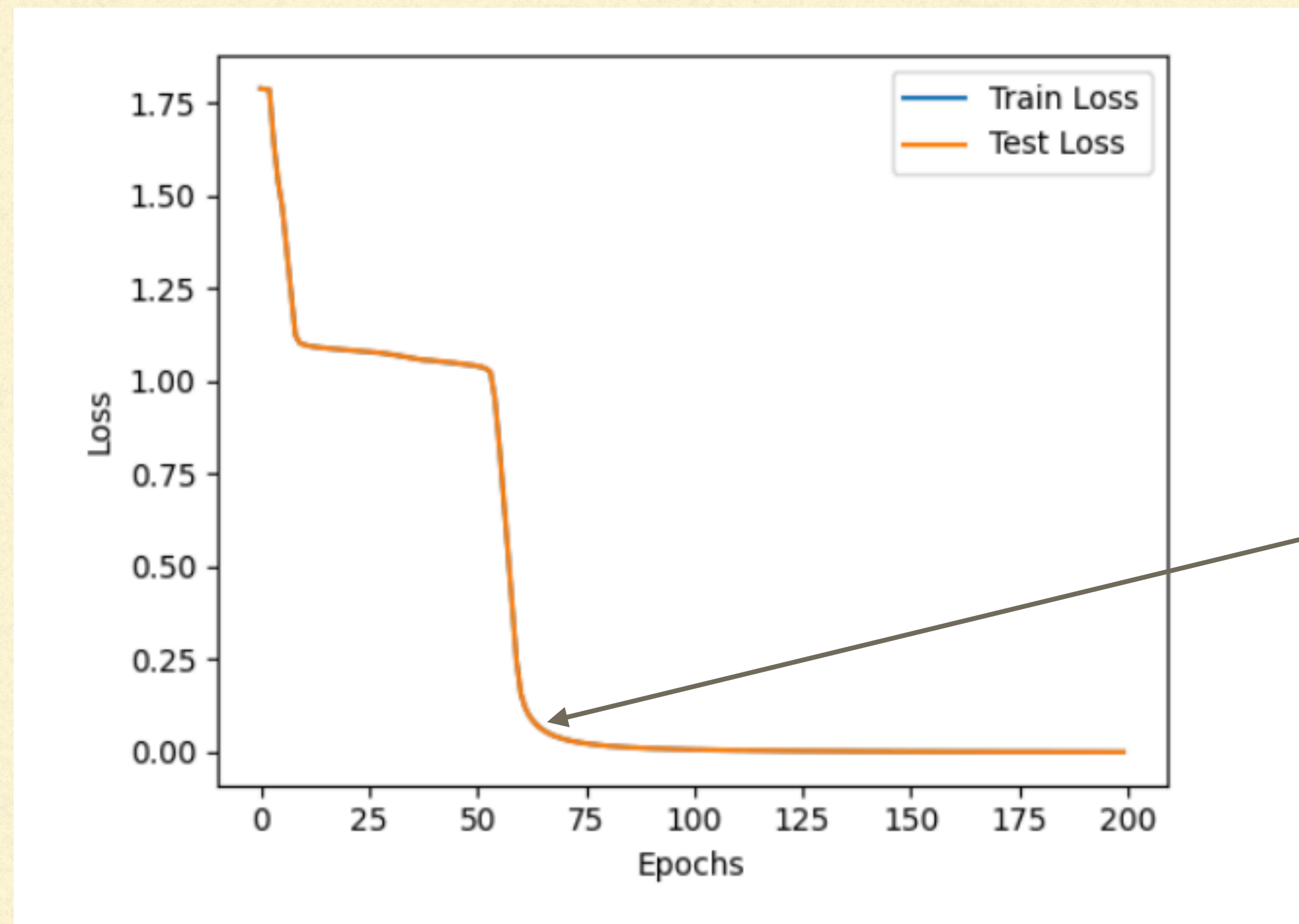


RECURRENT NEURAL NETWORK:



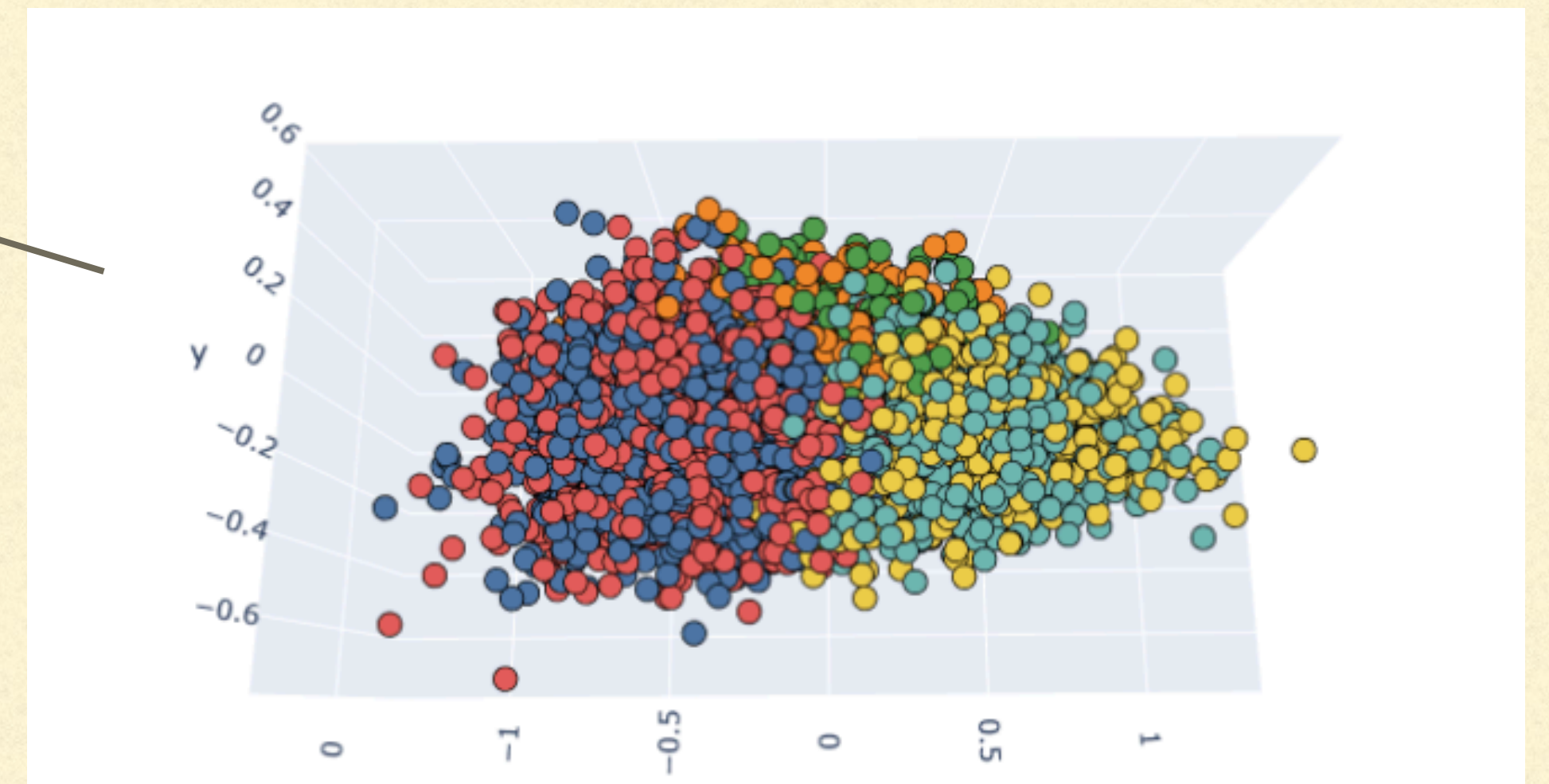
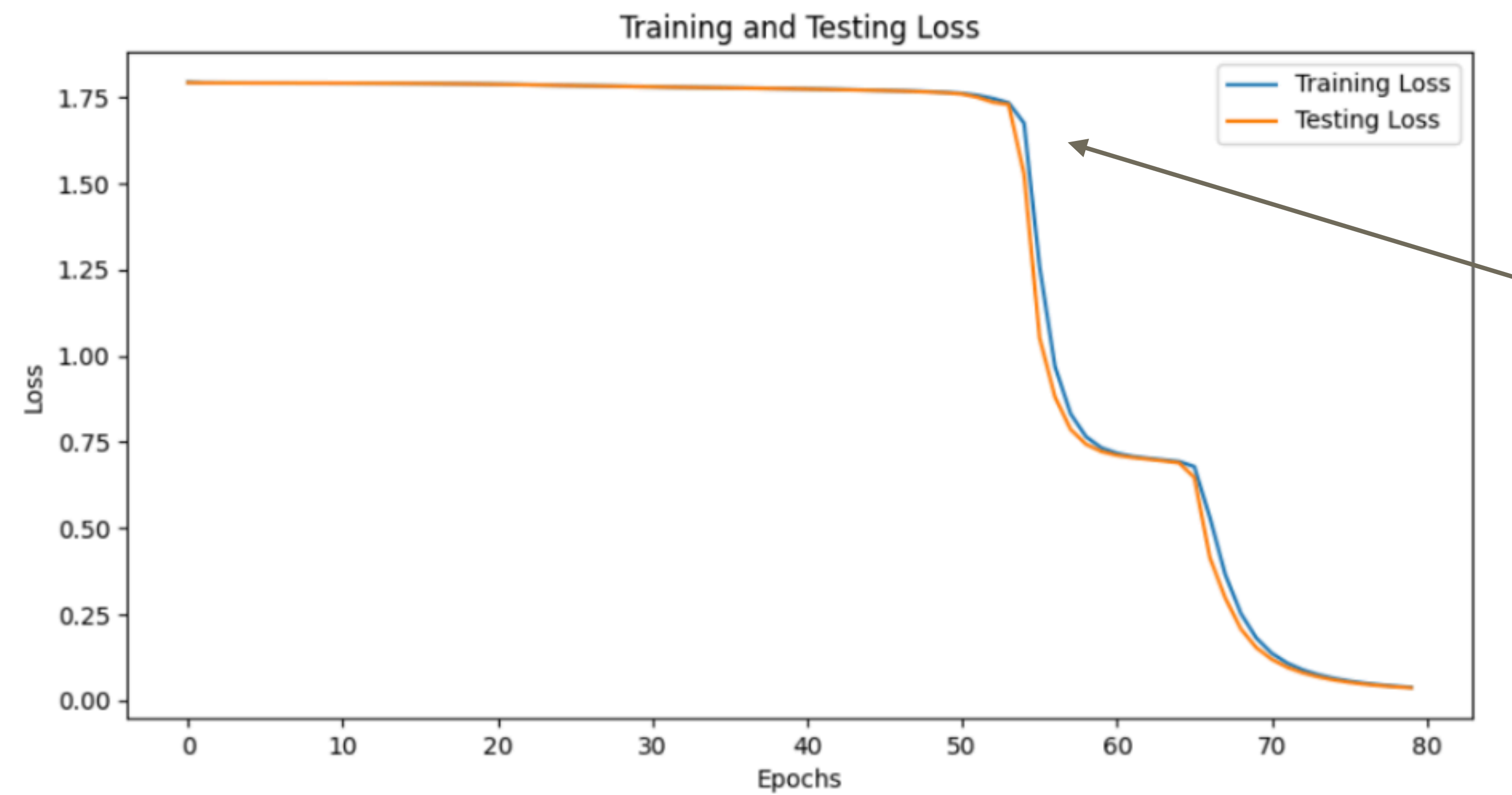
Model learned parity (natural quotient group) on the way to learning full group multiplication

RECURRENT NEURAL NETWORK:

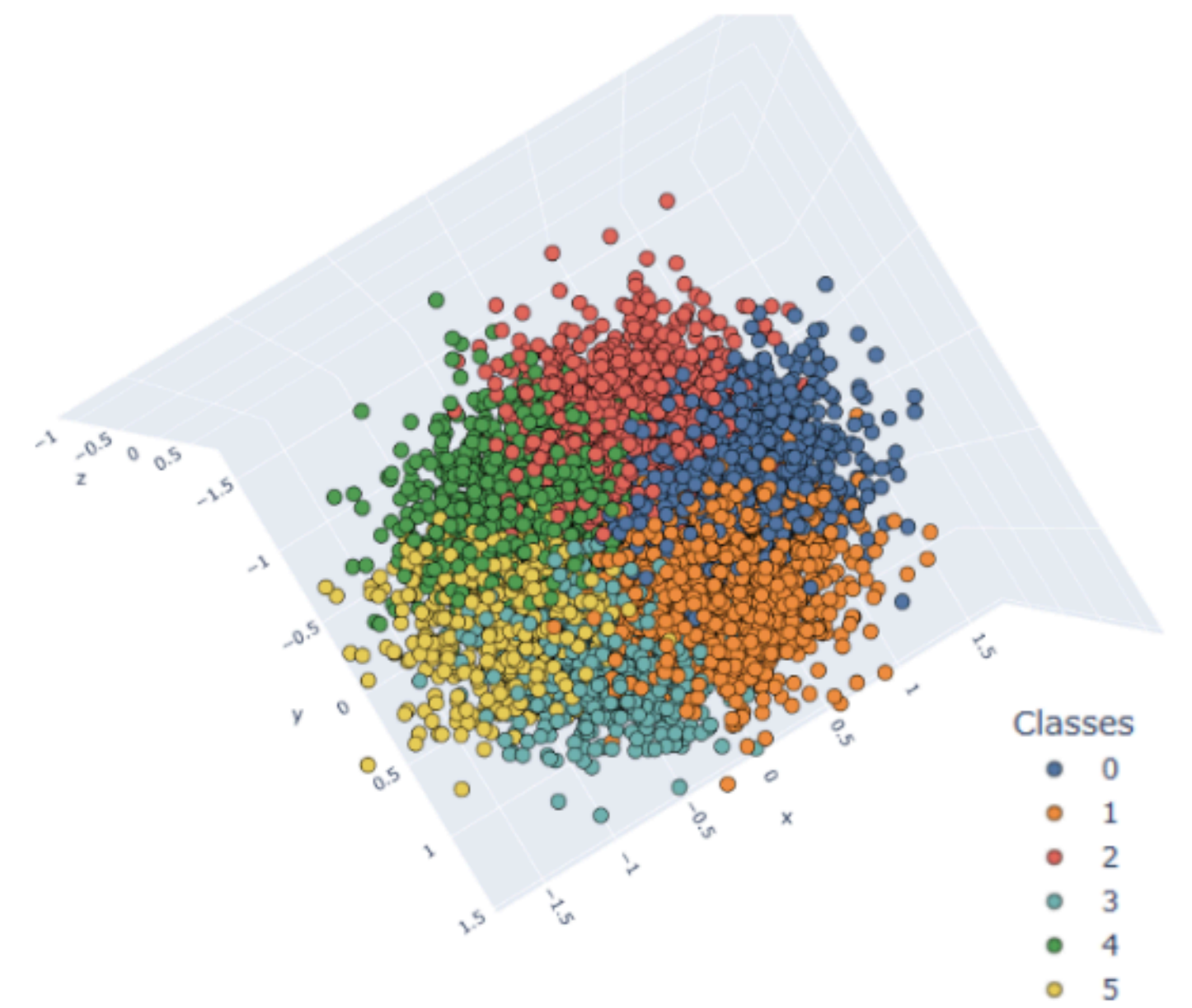
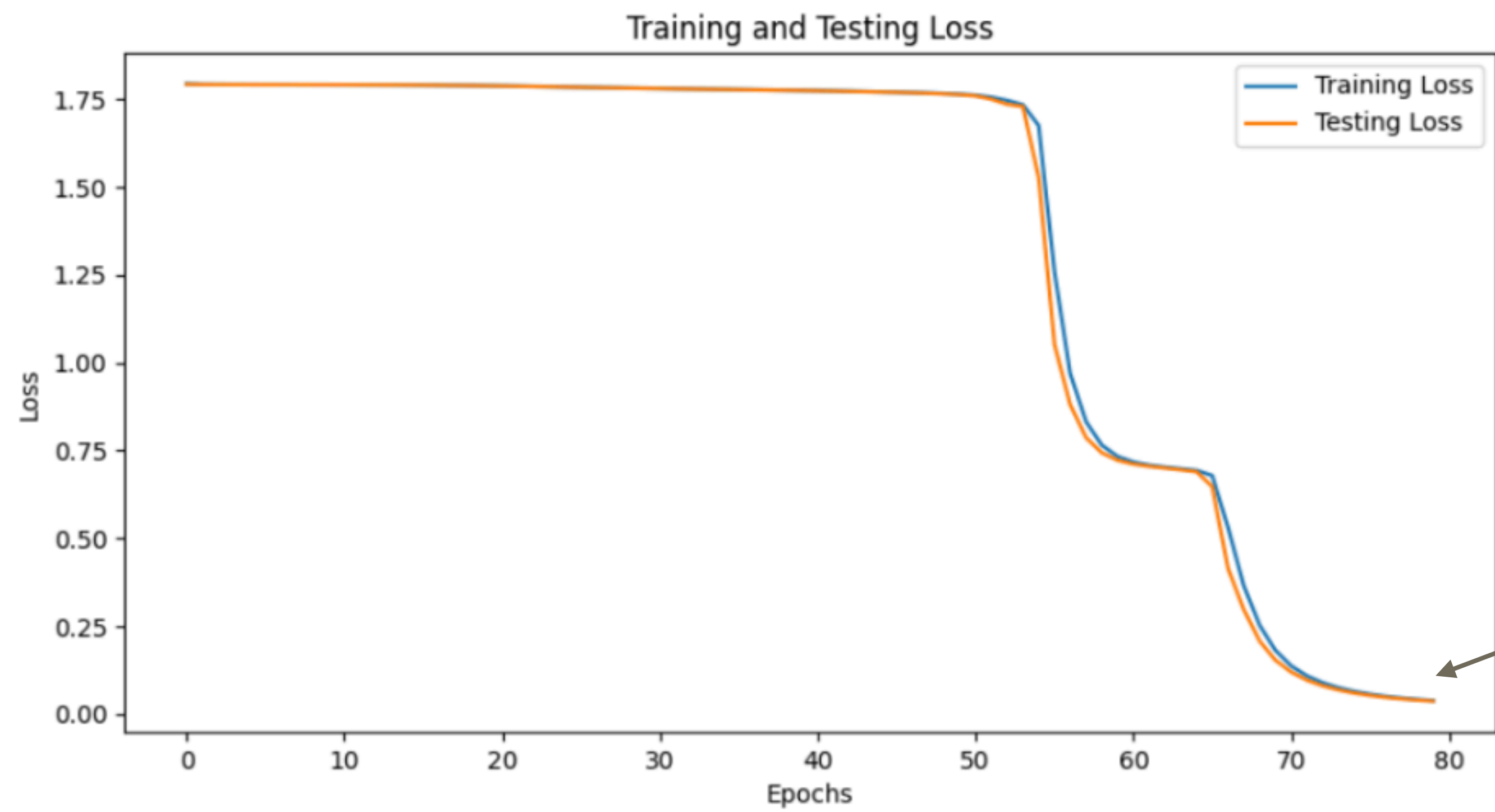


Model learned parity (natural quotient group) on the way to learning full group multiplication

LSTM:

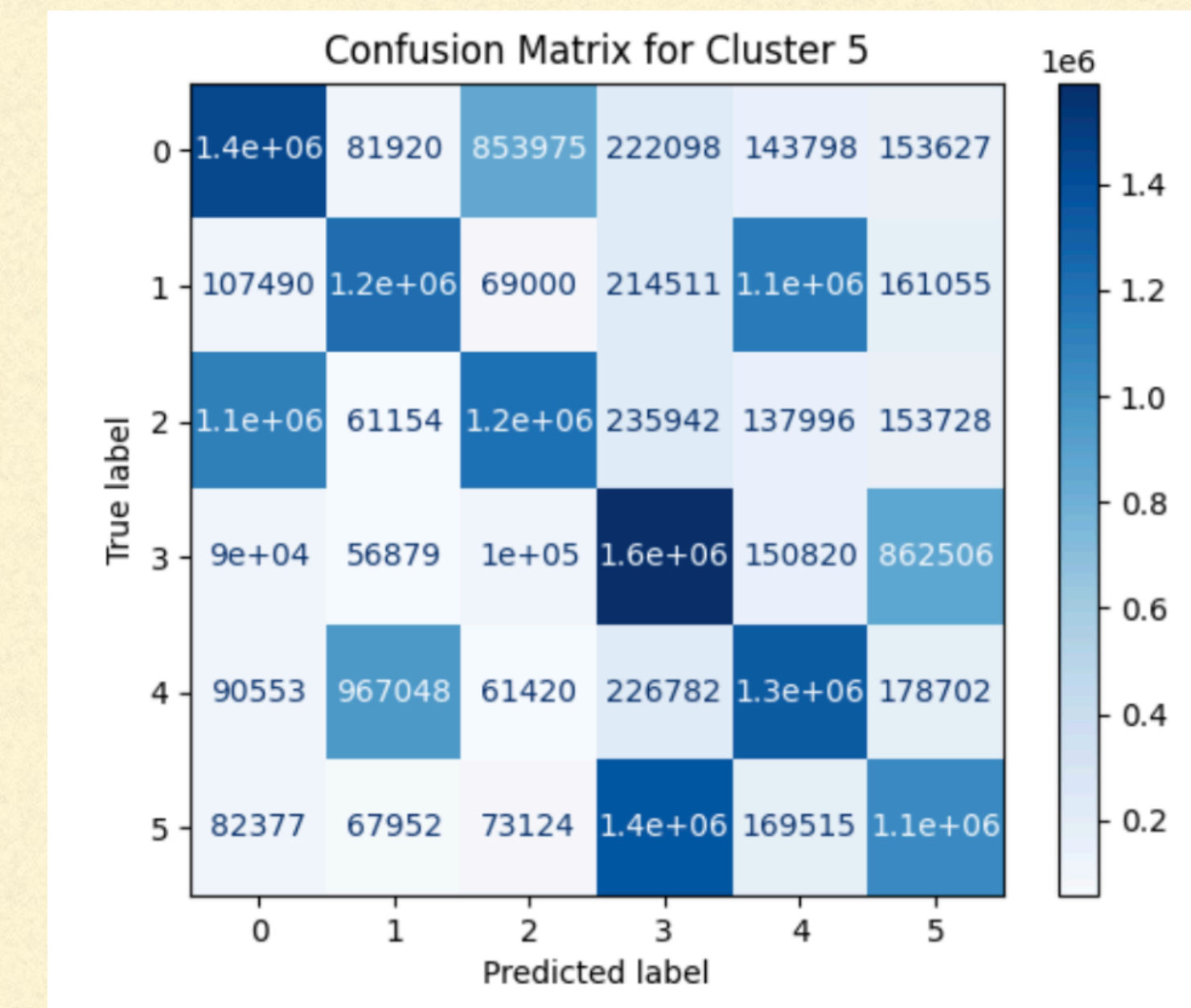
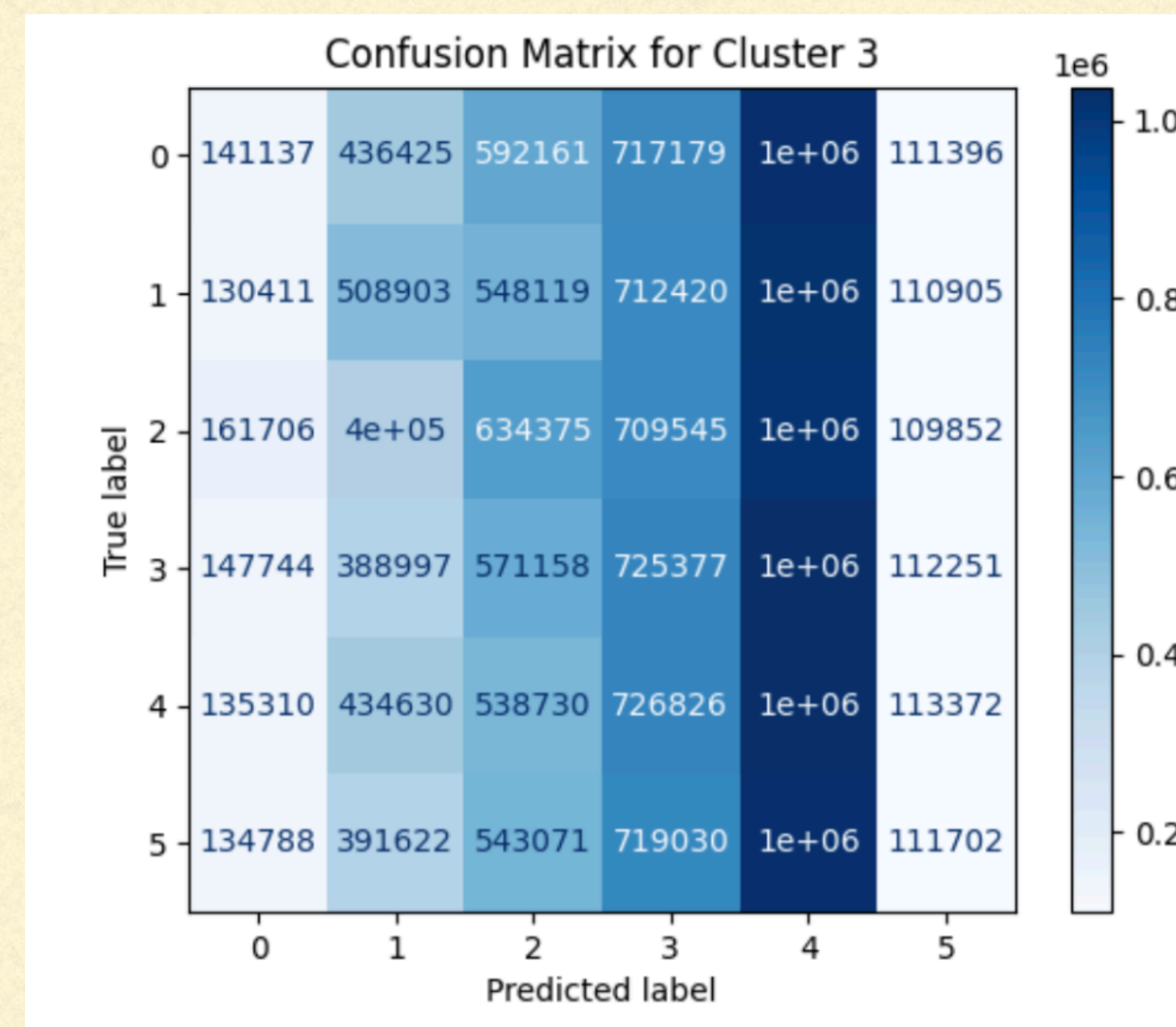
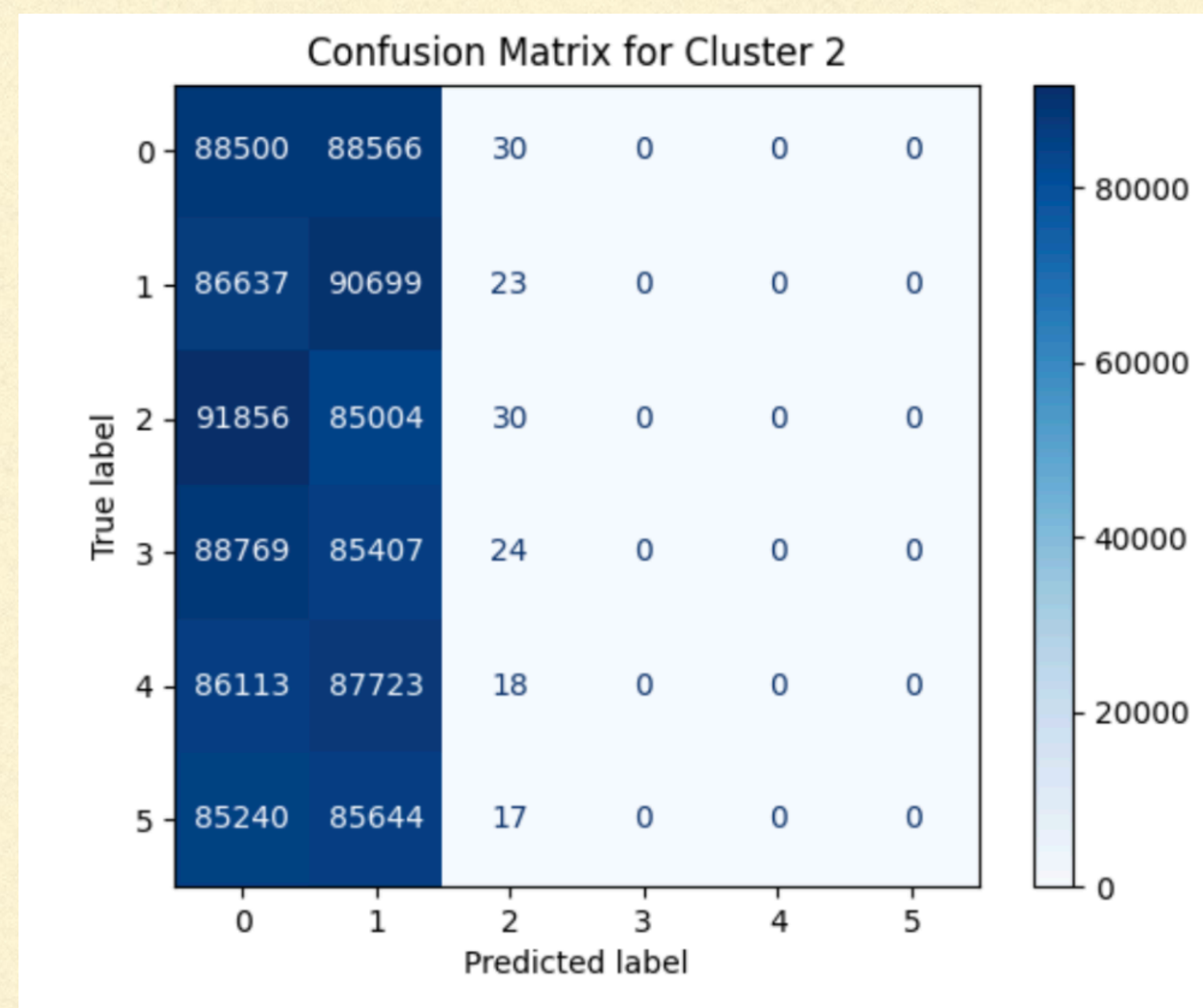


LSTM:



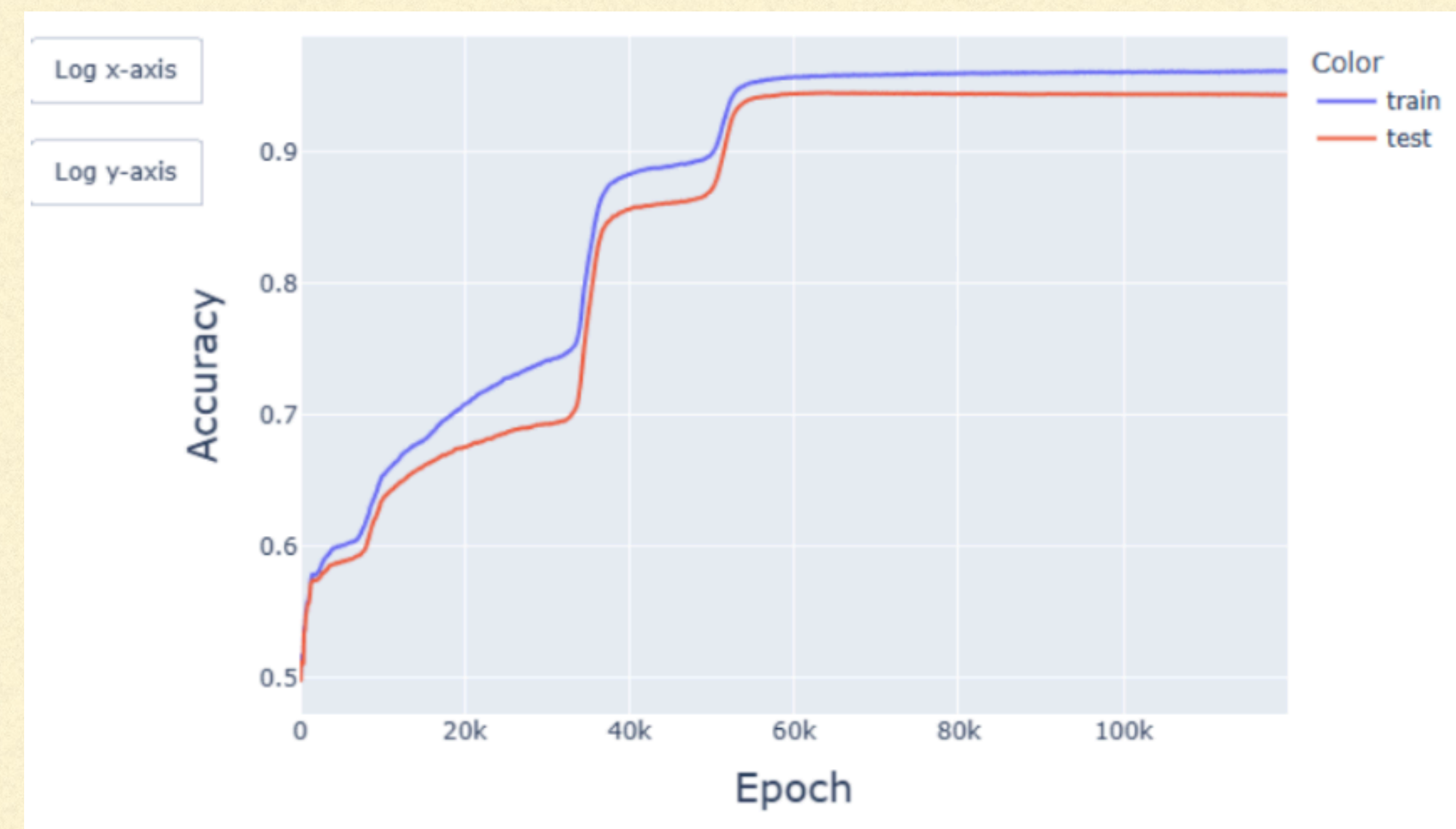
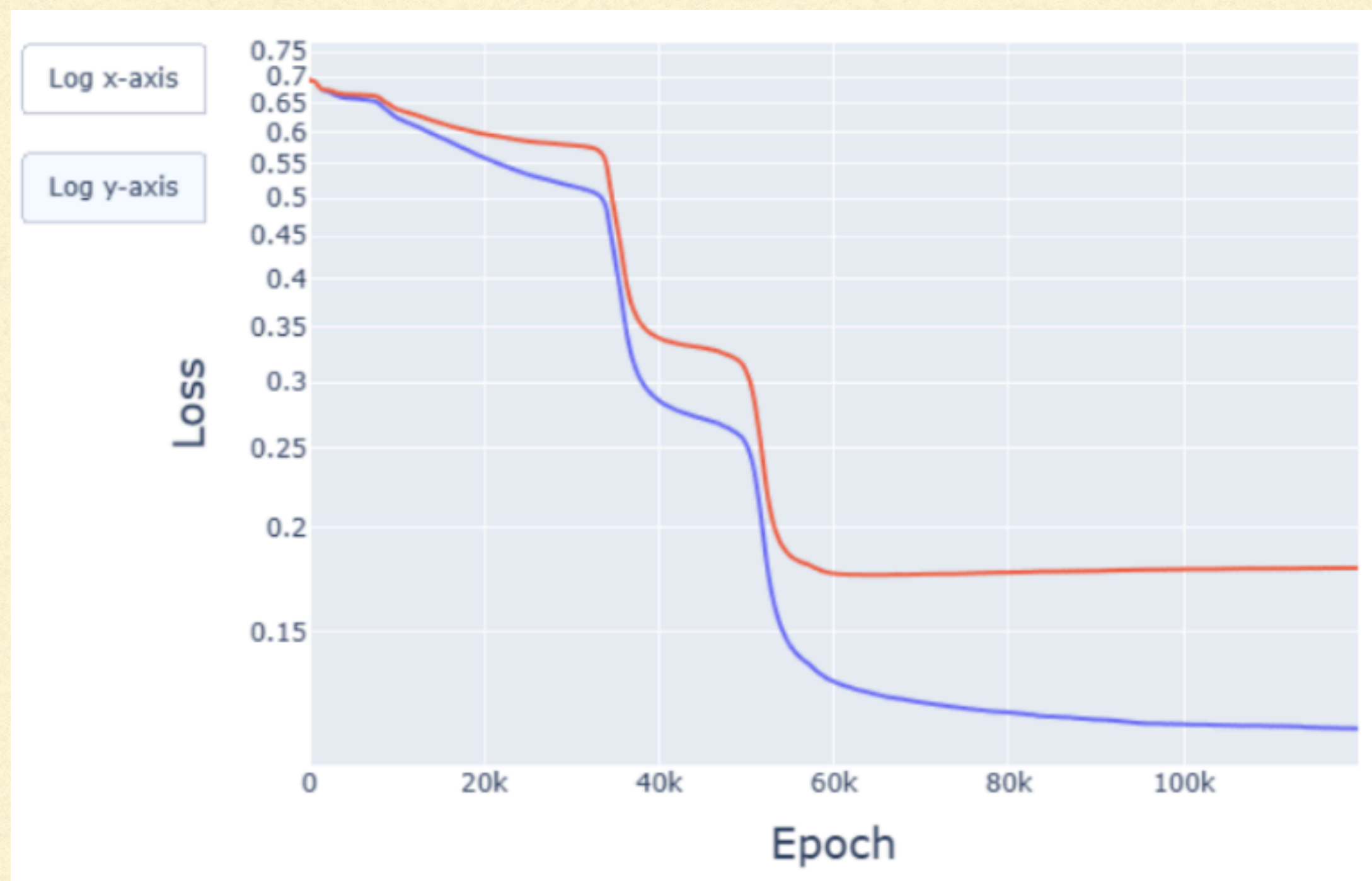
LSTM CONFUSION CLUSTERS:

(Clustered by type of confusion, roughly tracks Early, Mid, Late epochs of training)



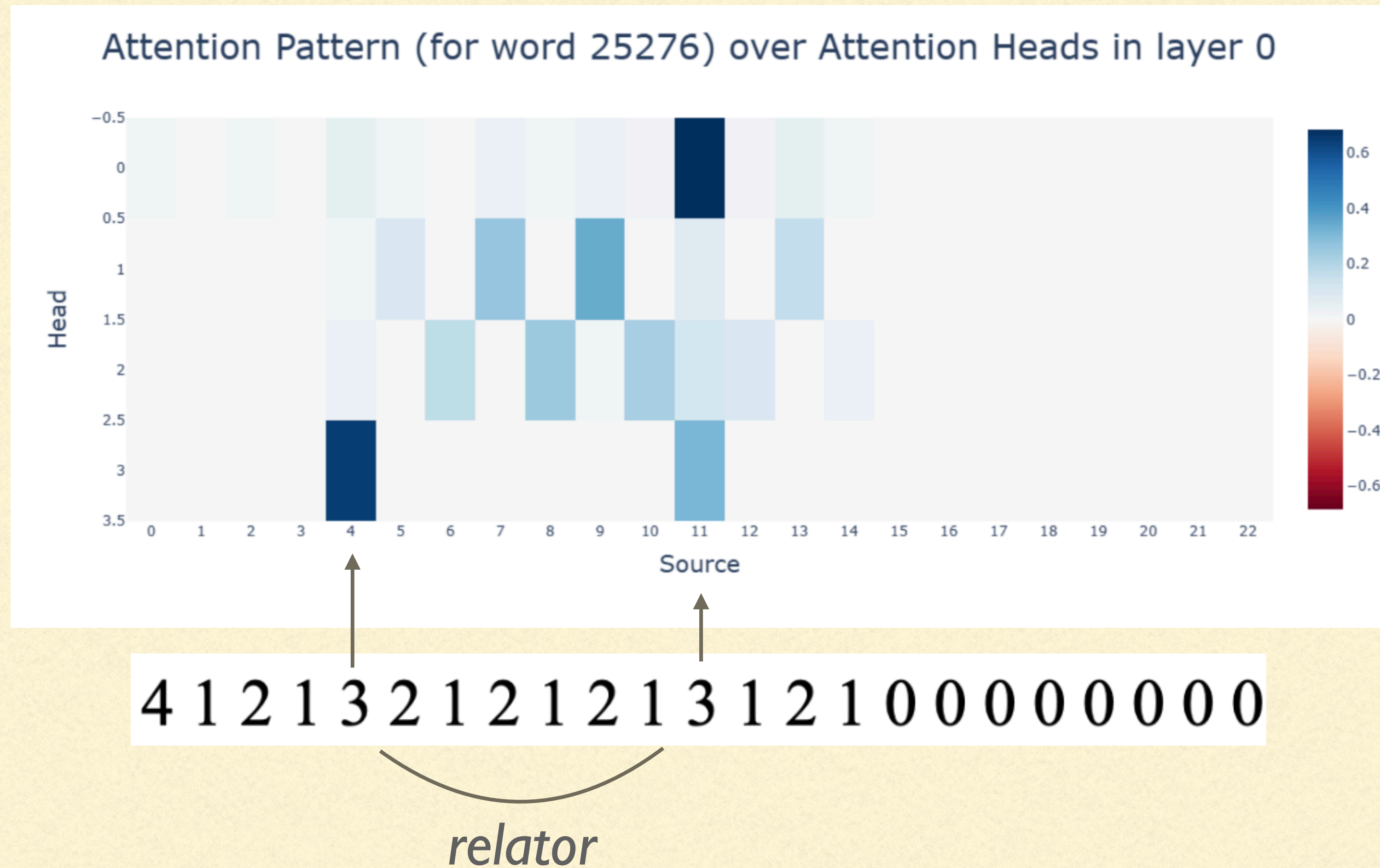
TRANSFORMERS:

Type 1 $\tilde{A}_2$ data set



Training was much more computationally expensive for transformer than other models

TRANSFORMERS:



TAKEAWAYS:

- RNNs/LSTMs may be just as good at (if not better than) transformers at learning geometric information
 - Transformers may be better at noticing word-theoretic/combinatorial patterns via their attention heads
 - This type of work really benefits from having a big team
 - Interpretability research is really in its infancy, and is a good place for mathematicians to be involved on both sides
-

THANK YOU FOR BEING HERE
