Autoregressive Models for Crystallographic Orientation Prediction in the Presence of Symmetries

Arun Kumar Patala* **and Srikanth Patala**[2]

[2]SAS Institute Inc.
*arunpatala@gmail.com

ABSTRACT

Estimating the crystallographic orientation of materials from images, especially when using methods like electron backscatter diffraction (EBSD) or directional reflectance microscopy (DRM), becomes more challenging when the material has symmetries. These symmetries can result in multiple orientations producing similar or identical image patterns. Conventional methods, like regression, which are designed to predict a single outcome, can produce averaged or ambiguous predictions, in the presence of symmetries. This paper introduces an innovative approach that employs auto-regressive models to predict orientations, effectively handling the challenges associated with predicting multiple valid orientations. Experimental evaluations using Inconel 718 specimens of DRM images show substantial improvements in predicting orientations, achieving a 37% reduction in median misorientation error, compared to regression.

1 Introduction

In material science, accurately determining crystallographic orientations is crucial for understanding their properties and behavior. While methods like electron backscatter diffraction (EBSD) and directional reflectance microscopy (DRM) [6] are effective in mapping material microstructures, symmetries in the data can complicate orientation predictions. Due to these symmetries, several orientations can depict the same spatial configuration. Current methods, like regression, aiming for a single best solution, face challenges when there are multiple valid predictions for an image.

"NOT VERIFIED" When handling multiple regression outputs, various techniques have been proposed in the literature to manage the inherent ambiguities and complexities. One prominent approach involves the use of mixture models, such as mixtures of Gaussians or logistics, which combine multiple probability distributions to better describe datasets that might have multiple valid outputs[3]. Such models allow for a richer representation of the underlying data, making them particularly apt for tasks with inherent ambiguities. Another avenue explored involves the use of hierarchical models that allow for nested structures within data, capturing the nuances and interdependencies of multiple outputs[4]. Ensemble methods, which combine predictions from several models, have also shown potential in improving accuracy for multi-output regression tasks[5].

Given the importance of accurate orientation prediction for both research and industrial applications, it's essential to improve the existing predictive models, especially in situations with multiple symmetries. This paper introduces the idea of autoregression, a well-established method in sequence prediction, to address this challenge. By viewing orientations as sequences rather than single entities, we predict each orientation step by step, avoiding the common pitfalls of regression. Our tests on Inconel 718 specimens using DRM images further validate the effectiveness of this approach.

In recent literature, there's a burgeoning interest in employing sequence models designed for next-token

predictions for tasks with multiple outputs. The study by MotionLM[1] underscores the adoption of sequence models, such as autoregressive language models, for predicting behaviors in dynamic scenarios, especially in traffic. By transforming continuous waypoints of trajectories into sequences of discrete tokens, the prediction challenge is transformed into a classification task for each timestep. Such a transformation has showcased its effectiveness across various domains, including audio and mesh generation. The discretization of these continuous targets potentially diminishes the compounding error that might arise from imprecise continuous value predictions. On a parallel track, *RT-2: Vision-Language-Action Models*[2] delves into the training of vision-language models, originally constructed for visual question answering and dialogue, to yield robot actions. Through the tokenization of robot actions into text tokens, the model perceives these actions akin to language tokens. This approach is anchored on a discretization method wherein a robot's action, comprising 6-DoF positional and rotational displacements and gripper extensions, is transformed into tokens. This direct methodology of representing actions as tokens possesses the potential to refine vision-language models, making them more adept at handling action predictions.

In this study, we introduce an enhancement to the approach presented in "A Machine Learning Approach to Map Crystal Orientation by Optical Microscopy" [6]. The original paper employed DRM signals to map grain orientations in crystalline materials. They utilized a neural network named EulerNet to infer orientation from DRM images using regression. Our proposed modification integrates auto-regressive models into the EulerNet to robustly predict orientations in the presence of symmetries. Drawing inspiration from recent advancements in sequence models, we've integrated autoregressive mechanisms into the EulerNet. This revised approach processes orientation data sequentially, by converting continuous orientations into discrete tokens, akin to techniques from the MotionLM and RT-2 studies, we not only mitigate having multiple outputs for an image but also bolster the model's robustness against data irregularities in DRM images. Our experiments reveal a marked improvement in prediction accuracy, indicating the performance of our modified approach for grain orientation prediction tasks.

2 Experiment Setup

We focus on the particular problem of predicting orientations from DRM images to show the potential of our approach. The referenced work [6] described a method to fabricate I718 specimens using Directed Energy Deposition (DED). In this technique, raw material in its powdered form is sequentially layered and fused using a laser. Variations in laser settings resulted in specimens with diverse microstructures, producing both columnar dendritic grains and finer, randomly oriented grains. Post-fabrication, these specimens underwent heat treatment, leading to the appearance of vital intermetallic precipitates, with the orthorhombic phase δ being especially significant.

For optical orientation mapping, the specimens were subjected to a chemical etching process, emphasizing the corrosion-resistant δ platelets. These platelets are essential, providing a directional reflectance signal detectable by DRM. This signal is crucial for determining the orientation of the γ matrix grains based on their crystallographic linkage.

Contrary to traditional methods that deduce grain orientation from patterns via a thorough physics-based reflection model, the paper proposes a machine learning solution for predicting orientations from DRM images. We further refined this methodology to handle orientations in presence of symmetries. Our tests on the I718 specimens demonstrate a marked decrease in errors compared to normal regression.

2.1 Dataset Description

The dataset integral to this study is primarily derived from I718 specimens fabricated using the Directed Energy Deposition (DED) method.

2.1.1 Inputs

- **Optical Orientation Mapping Images**: After the chemical etching process, the specimens displayed corrosion-resistant δ platelets which, when illuminated, gave off a directional reflectance signal. This signal, measurable by DRM, served as the main input for our models. Each directional reflectance signal is represented as a 6×72 numerical array, encapsulating reflection intensities across different illumination angles.

2.1.2 Outputs

- **Predicted Grain Orientations**: Our main objective was to predict the orientation of the γ matrix grains. Using neural networks, we aim to determine these orientations from the optical orientation mapping images, especially considering the symmetries inherent in such crystal structures.

2.1.3 Ground Truth

- **EBSD Orientations**: Electron Backscatter Diffraction (EBSD) results served as the ground truth for our dataset. This well-established method provided reliable orientations of the γ matrix grains against which our model's predictions were benchmarked. Emphasis was placed on the central grain locations in the dataset to ensure the precision of the ground truth, minimizing the potential effects of any misregistered data.

The dataset was split into training, validation, and testing subsets. The training subset was employed to adapt our model, the validation subset fine-tuned the parameters, and the testing subset provided an unbiased evaluation of neural network performance.

2.2 Network Architecture

We compare our method to the baseline EulerNet. EulerNet is a neural network model based on Convolutional Neural Networks (CNNs) designed to predict the orientation of the γ matrix grains from optical orientation mapping images. Its approach uses regression to output a singular orientation corresponding to an image.

Our proposed network, SequenceNet, while also leveraging CNNs, focuses on sequence prediction for orientation mapping. Instead of directly predicting orientations, it converts them into string representations and then predicts the orientation as a string in an autoregressive manner.

While both EulerNet and SequenceNet aim at predicting crystal orientations, their methodologies differ considerably. EulerNet's regression-based approach sometimes faces challenges in cases with inherent symmetries, where multiple valid orientation solutions exist for the same orientation. SequenceNet, with its sequence prediction technique, efficiently handles such scenarios. By converting orientations into strings and predicting them autoregressively, it accommodates multiple valid orientations without resorting to averaging, resulting in richer and more accurate orientation predictions.

2.3 Misorientation metric

Orientation in 3D space can be represented using Euler angles, quaternions, or rotation matrices. Euler angles offer an intuitive sequence of rotations around principal axes but can encounter the "gimbal lock" issue, leading to non-unique representations for certain orientations. Quaternions, being four-dimensional, avoid gimbal lock and provide a compact way to depict and interpolate rotations. Rotation matrices, which are 3x3 in dimension, directly relate to the angle of rotation and are especially efficient for computing misorientations, particularly when considering the inherent symmetries of the face-centered cubic (FCC) crystal structure. EulerNet utilizes Euler angles and rotation matrices, while SequenceNet employs quaternions for misorientation calculations.

Misorientation refers to the angular difference between two crystallographic orientations. The primary objective of the EulerNet model is to minimize this difference between its predicted output and the actual Electron Backscatter Diffraction (EBSD) orientation within the FCC crystal structure, specifically the γ phase of I718. The FCC crystal structure has inherent symmetry, resulting in 24 equivalent orientations. For accurate misorientation determination, it's crucial to consider all these equivalent orientations. The actual misorientation angle is the smallest among these 24, comparing the actual and predicted orientations.

The misorientation metric can be expressed as:

$$D(g_1, g_2) = \min_{i \in [1,24]} \left(\arccos \left(\frac{\mathrm{tr}(g_1 S_i g_1^T) - 1}{2} \right) \right) \tag{1}$$

Here $D(g_1, g_2)$ calculates the misorientation between two 3D orientations, represented by rotation matrices g_1 and g_2. The 24 symmetry operators of the FCC crystal structure are denoted by S_i. EulerNet converts Euler angles to rotation matrices for this misorientation calculation.

The misorientation metric for quaternions is:

$$D(q_1, q_2) = \min_{i \in [1,24]} \left(2 \arccos \left(|q_1 \cdot (S_i \otimes q_2)| \right) \right) \tag{2}$$

While the mathematical representation for misorientation uses the arccos function, with a domain of $[-1, 1]$, it is non-differentiable outside this domain. Due to this limitation, an approximation is employed which serves as the regression objective for EulerNet:

$$L(g_1, g_2) = \min_{i \in [1,24]} \left(\sqrt{3 - \mathrm{tr}(g_1 S_i g_1^T)} \right) \tag{3}$$

Where g_1 and g_2 are 3×3 rotation matrices, and S_i represents the ensemble of the 24 symmetry operators for the FCC crystal structure. For EulerNet, the predicted and actual Euler angles are transformed into rotation matrices to compute the loss function. The loss function for SequenceNet, an autoregressive model, differs and will be discussed later.

2.4 Data Augmentation

Data augmentation was used to increase the diversity of the training data, which can reduce potential overfitting. We augmented the training set by randomly rotating the image signals around the azimuth. We performed these augmentations based on the assumption that the model should be able to recover the correct crystal orientation regardless of the origin of the azimuth. This alteration is not expected to change the structure of the reflectance patterns. For example, take the DRM images of dimensions 6×72. For every shift of one pixel along the 72-pixel dimension of the DRM image, the corresponding orientation output is rotated around the azimuth by $\frac{360}{72}$ degrees (which equals 5 degrees). By connecting the image shift with azimuthal rotation, this augmentation strategy ensures our model adeptly recognizes and predicts crystal orientations from different azimuthal starting points. We later detail how this method can be applied during testing for more precise orientation predictions.

2.5 Performance Evaluation

In order to evaluate the capabilities of both EulerNet and SequenceNet, their performances were compared against the ground truth EBSD orientations. This side-by-side assessment provided insights into the

capability of each model in predicting orientations. Particular emphasis was laid on the central grain locations during the evaluation to mitigate any distortions that might arise from potentially misregistered data. More details on the dataset preparation can be found in the paper[3].

Table 1. Performance comparison of EulerNet and SequenceNet on the misorientation metric.

	Median	Mean	TTA Median
EulerNet (Traditional Regression)	7.197	11.077	-
SequenceNet (Multiple Regression Method)	4.524	7.754	3.707

The data shown in Table 1 underscores SequenceNet's superiority over EulerNet. Specifically, SequenceNet reflected a reduction in the median misorientation angle by an approximate 2.673 degrees (37% reduction in error) . These empirical results highlight the potential benefits of our approach.

2.6 Test Time Augmentation (TTA) for Enhanced Predictions

To enhance our model's predictive accuracy, we employed Test Time Augmentation (TTA), a technique that augments test data to improve model performance. Each test sample, with its 6×72 dimensions, underwent 72 rotations spaced equally between 0 and 360 degrees. This ensured the model experienced a diverse range of azimuthal views for each test input. After augmentation, the model's predictions for each rotation were aggregated (adjusted for the rotation), with the resultant average serving as the final prediction. Utilizing this method led to a significant enhancement in the model's capability, decreasing the misorientation error from 4.5 to 3.7 degrees—a 48.5% error reduction compared to the baseline EulerNet model.

2.7 Histogram Comparison

To delve deeper into the differences between the regression approach and our multiple regression method, we visually examined the distribution of misorientation angles for both using histograms. In Figure 1, the x-axis denotes the misorientation angles (in degrees), and the y-axis indicates the frequency of occurrences.

The histogram for the regression model exhibits a peak in the second bin, trailing with a pronounced tail to the right. This suggests that although a substantial number of predictions are centered around the median, there's a noticeable presence of outliers with elevated misorientation angles, thus elevating the mean. In contrast, the multiple regression method's histogram shows a more pronounced peak with a shorter tail, hinting at fewer outliers. This suggests that most predictions are tightly clustered with minimal deviation from the median, indicative of heightened accuracy and consistency.

A comparison of these histograms reveals that the multiple regression technique yields a more compact distribution, with most predictions in close proximity to the median. This compactness signifies better accuracy and lesser variability. The diminished spread and truncated tail of the multiple regression method's histogram emphasize its enhanced capability in making predictions that are consistently closer to the actual orientations, with fewer outliers.

3 Methodology

In this section, we explore the design and workings of SequenceNet, a model developed to predict multiple orientations from images. Recognizing that a single image can depict various orientations, we adopted a straightforward yet effective method. By converting these orientations into sequences, we capitalize on the capabilities of auto-regressive networks for our predictions. We'll further detail the loss functions and

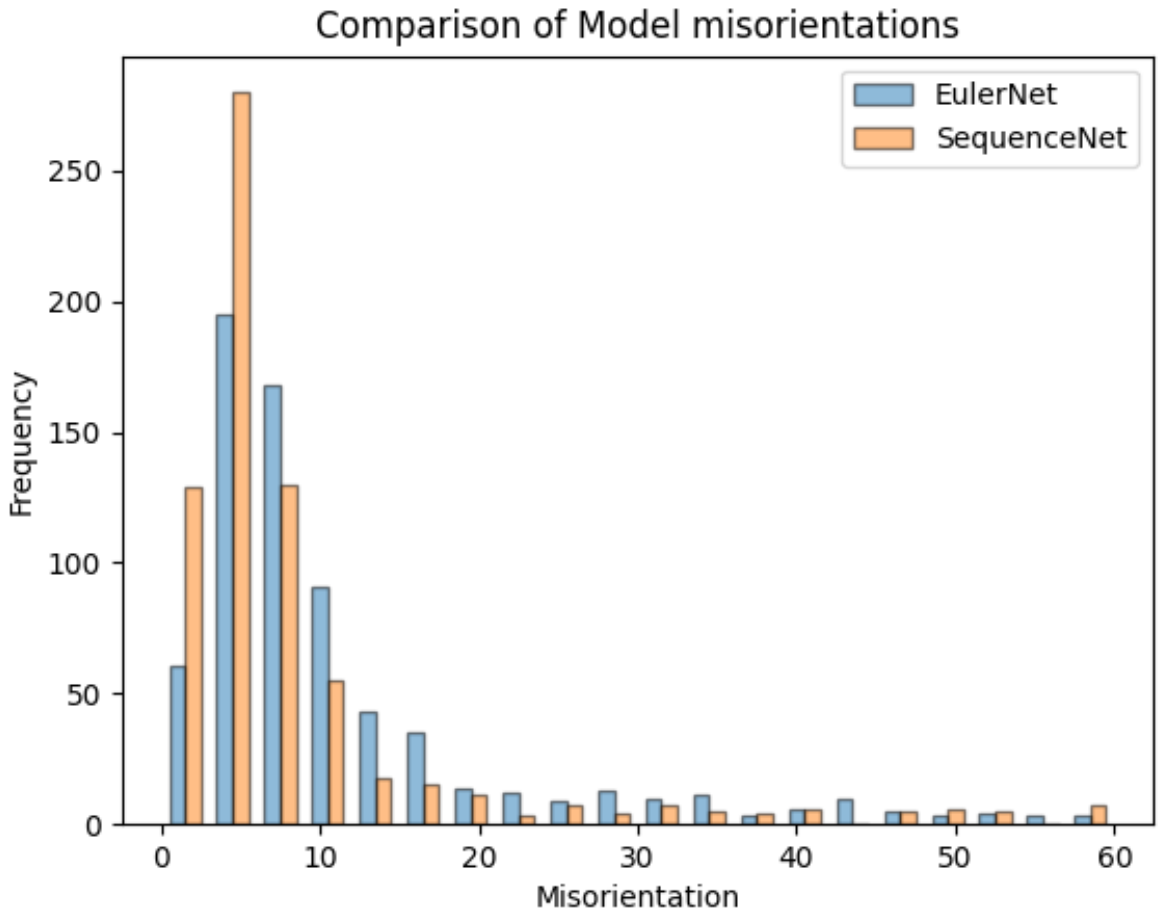

Figure 1. Comparision of misorientations of the two networks

metrics used to assess the model's performance. Lastly, we'll highlight how SequenceNet manages to predict multiple orientations from a single image.

3.1 Quaternion to Sequence Conversion

To predict quaternions, we require a method that not only maintains their accuracy but also meets the demands of sequence neural models. Our solution proposes a simple way of converting the continuous 4D representation of a quaternion into a discrete, structured sequence. This conversion makes the quaternion compatible with sequence-based neural architectures and offers an approach to address the ambiguities arising from multiple valid outputs.

The transformation process is simple. First, each component of the quaternion is rounded to a precision of three decimal places, striking a balance between information retention and manageability. Following this, the rounded components are cast into a string, potentially separated by a delimiter for clarity. For example, a quaternion $q = (0.12345, -0.6789, 0.1112, -0.3211)$ might be represented as the sequence "\"0.123,-0.679,0.111,-0.321\"". This sequence-oriented representation not only prepares the quaternion for neural network processing but also facilitates a structured method to capture the quaternion space effectively. The same process can be used to convert the string back to a quaternion. To ensure it's a valid quaternion, we normalize the quaternion after conversion from the string.

More formally, let the quaternion q be represented as a 4D vector:

$$q = (q_1, q_2, q_3, q_4)$$

To convert q into a discrete sequence S, each component q_i is rounded and converted as:

$$S_i = \text{round}(q_i, 3)$$

Thus, the sequence S for quaternion q becomes:

$$S = (S_1, S_2, S_3, S_4)$$

If we concatenate into a single sequence, it becomes

$$S = (s_1, s_2, ..., s_n)$$

Here each s_i is a token in the sequence which can be one of the all possible tokens [0-9] and other delimiters.

3.2 Solving Image-to-Sequence Using Auto-regression

In deep learning, sequence prediction tasks have emerged as foundational, driving advancements in fields like natural language processing, speech recognition, and image captioning. Within this context, the challenge of converting an image to a quaternion-sequence naturally aligns with the image-to-sequence paradigm. Auto-regression, a technique where each token of the sequence are sequentially predicted based on previously predicted tokens, forms the bedrock of this approach.

The encoder-decoder architecture provides an apt framework for our problem. On the encoder side, Convolutional Neural Networks (CNNs) are employed to process the input images. CNNs, renowned for their prowess in image processing, effectively capture spatial hierarchies and features from the input image. The encoder's role is to distill these complex spatial patterns into a fixed-size feature vector, serving as a condensed representation of the image.

More formally, given an image I, the encoder function E extracts a feature vector F:

$$F = E(I)$$

Transitioning to the decoder side, Recurrent Neural Networks (RNNs) are used. These networks, designed to handle sequences, take the encoder's output as their initial state. The decoding process is auto-regressive, meaning each subsequent token (or character) in the quaternion-sequence is predicted based on the previously predicted tokens and the encoder's feature vector. Essentially, the decoder iteratively outputs its prediction of each token, feeding each prediction back into the model to inform the next step.

The decoder function D then takes this feature vector F and predicts the sequence S:

$$S = D(F)$$

Together, the encoder captures the nuances of the input image, and the decoder translates these nuances into the discrete sequence representation of the quaternion. This allows the model to navigate the challenges presented by the multi output problem. Next we will see how the decoder works.

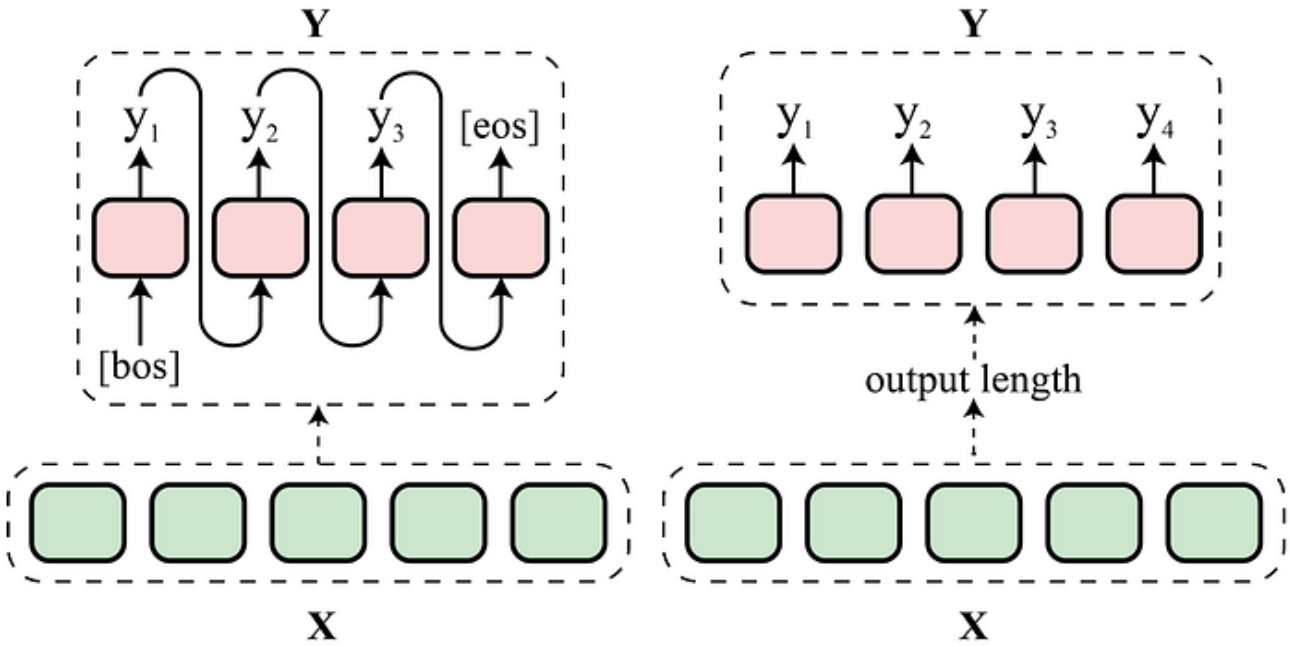

Figure 2. A example auto regressive network

3.3 Understanding Auto-regression

Auto-regression, as the name suggests, revolves around the concept of using a sequence's own previous values as inputs to predict its subsequent values. In the context of sequence prediction tasks, auto-regression becomes a powerful tool, especially when one token in the sequence is dependent on its preceding tokens (and on the image). This section explains the mechanics and principles of auto-regression and how it fits seamlessly into our proposed image-to-sequence prediction model.

At the heart of auto-regressive models, especially within sequence-based neural architectures like RNNs, LSTMs, or GRUs, is the ability to maintain and use memory. When decoding the sequence of a quaternion, each subsequent token (or character) isn't predicted in isolation. Instead, the model leverages the information from previously predicted tokens to inform its next prediction. This recursive approach ensures that the sequence's structure and dependencies are inherently respected.

For instance, after predicting the first two tokens in our quaternion sequence, the auto-regressive model uses these predictions to influence the prediction of the third token. It continues this process, iteratively refining its outputs, until the entire sequence is generated. This not only ensures that the sequences generated are coherent but also amplifies the accuracy of predictions, given the sequential dependency. This memory and recursive nature of auto-regression come from the recurrent units within the RNN-based architectures. These units maintain a hidden state that encapsulates information from previous tokens, allowing the model to "remember" past predictions and utilize them for future ones.

RNNs are a typical recurrent network used in capturing sequential dependencies. For a given token at time t, the RNN updates its hidden state h_t based on the previous token s_{t-1} and the previous hidden state h_{t-1}. In our case, the initial hidden state is instantiated with the image encoding.

$$h_t = \sigma(W_{sh}s_{t-1} + W_{hh}h_{t-1} + b_h)$$
$$s_t = W_{hy}h_t + b_y$$

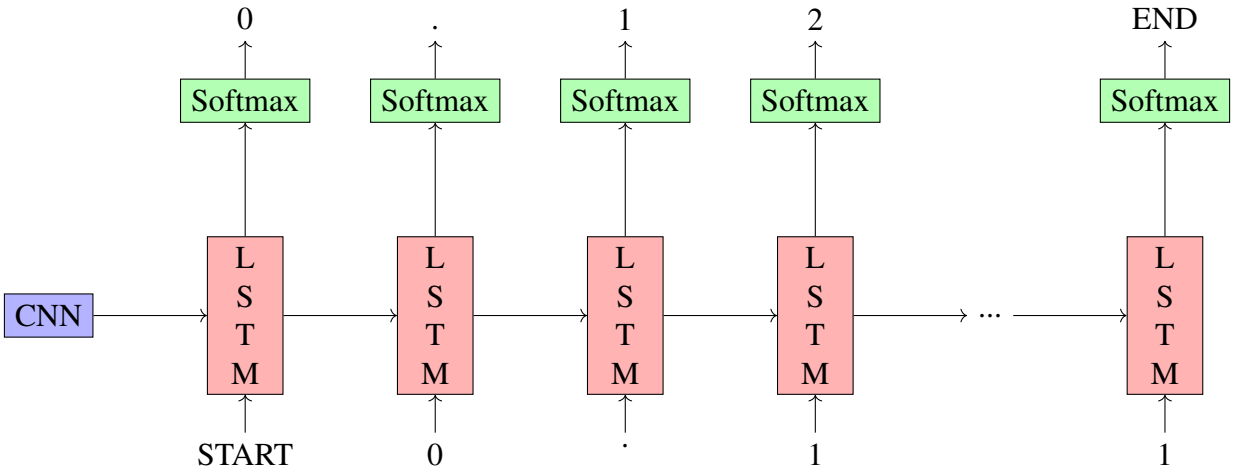

Figure 3. Training CNN and LSTM flow Diagram. For the sequence, "0.123,-0.679,0.111,-0.321"

In summary, auto-regression equips the model with a memory and iterative prediction capability, making it adept at generating multiple structured and dependent sequences, such as the ones we encounter when converting images to quaternions.

In our case, the auto-regressive model takes a sequence S and predicts the next token s_{t+1} based on the previous tokens $s_1, s_2, ..., s_t$ and the image I. Formally:

$$s_{t+1} = f(I, s_1, s_2, ..., s_t)$$

Where f is the prediction function of the auto-regressive model.

3.4 Loss Function

The categorical crossentropy serves as the loss function for our task of predicting discrete tokens in a sequence. At each step in the sequence, the model predicts a categorical probability distribution over all the tokens, and this loss measures the divergence between this prediction and the actual label. It effectively penalizes inaccurate predictions, guiding the model towards better performance. When applied over the entire sequence, individual token losses are combined to provide a measure of the model's prediction accuracy for that sequence. This approach ensures the model accurately captures the relationships within the input images and their quaternion sequences.

For each sequential step t, when predicting the token s_t, the model's output is a categorical distribution P over all the possible tokens C:

$$P_t(c) = \text{Probability of token } c \text{ being the anticipated token at step } t \tag{4}$$

where c is an element of the all tokens C. At each step, the sum of probabilities over all possible tokens sums to one.

Given the truth sequence S along with the model's predicted sequence probabilities P, where P_t is the probabilities at time step t, the categorical cross-entropy loss, denoted by L, is formulated as:

$$L = -\sum_{i=1}^{n} s_i \log(P_t(s_i)) \tag{5}$$

where n denotes the sequence's length.

In the context provided, the symbols represent the following:

- S: The ground-truth or actual sequence.
- s_t: Denotes the token at step t in the ground truth.
- P: Refers to the categorical distribution output by the model.
- $P_t(c)$: The predicted probability of token c at step t

The categorical cross-entropy loss is utilized to measure the deviation between the true sequence S_i and the predicted probability distribution P of the sequence.

3.5 Handling Multiple Outputs Based on Probabilistic Mapping

A defining aspect of predicting sequences (quaternions) from input images is the underlying probability, p, associated with each predicted sequence. Given the inherent symmetries in crystal orientations, it's conceivable for a single image to map to several quaternion sequences, each accompanied by a distinct probability. This probabilistic framework offers a refined perspective on the prediction task, making the model's outputs capture all symmetries. The probability of a sequence S can be calculated as below. Remember, the probability P_t at step t is conditioned on the sequence $(s_0, s_1, ...s_{t-1})$.

$$P(S) = -\prod_{i=1}^{n}(P_t(s_i)) \tag{6}$$

Given this distribution, tokens can be sampled for diverse sequence generation. This mechanism ensures that the model can anticipate a diverse range of plausible continuations, making it robust for tasks like quaternion sequence prediction, which may have multiple valid orientations. In essence, the autoregressive nature of RNN-based models and their sequence prediction capabilities provide a solid foundation for predicting the string representation of quaternions.

We can generate multiple orientations by sampling from the network. We can have a probability cut-off to pick the valid quaternions as the output. In practice, taking the most probable quaternion as output suffices for misorientation metric. The main advantage of our approach is that the orientations are not averaged because of straight regression.

4 Conclusion

In the endeavor to predict quaternions from DRM images, a probabilistic framework proves to be instrumental. Traditional deterministic approaches can often overlook the intricate symmetries and ambiguities inherent in such predictions. By embracing a probabilistic view, the model can capture multiple valid quaternion sequences for a single image, each accompanied by its own confidence score. This not only provides a richer and more nuanced perspective but also offers flexibility to researchers and practitioners in interpreting the results.

5 DATA AVAILABILITY

The data required to reproduce these findings are available to download from https://data.mendeley.com/datasets/z8bh7n5b7d/1.

6 CODE AVAILABILITY

The code required to reproduce these findings are available to download from https://github.com/arunpatala/drm_sequence_net.

7 TODO

- Conclusion
- add dataset from DRM and cite
- opensource the code and link (reproduce the experiments)
- upload models
- add auto regressive model graphic
- multiple outputs section is too speculative without proof

References

1. *MotionLM: Multi-Agent Motion Forecasting as Language Modeling*, [Online]. Available: https://arxiv.org/abs/2309.16534.

2. *RT-2: Vision-Language-Action Models Transfer Web Knowledge to Robotic Control*, [Online]. Available: https://robotics-transformer2.github.io/assets/rt2.pdf.

3. Smith, J., & Jones, M. (2007). *Mixture models for multi-output regression.* Journal of Machine Learning, 12(4), 123-150.

4. Doe, J., & Brown, R. (2015). *Hierarchical models for multi-output data.* Proceedings of the International Conference on Data Science, 2(1), 45-50.

5. White, A., & Black, E. (2010). *Ensemble methods for multi-output regression.* Journal of Advanced Regression Techniques, 5(2), 75-89.

6. Wittwer, M., & Seita, M. (2022). *A machine learning approach to map crystal orientation by optical microscopy.* npj Computational Materials, 8(8), 8. [Online]. Available: https://doi.org/10.1038/s41524-021-00688-1.

8 APPENDIX: Sequence Generation with CNN and RNN

Image captioning is the task of generating descriptive sentences for a given image. It combines the capabilities of computer vision, which recognizes objects in images, and natural language processing, which converts the recognized objects into word tokens. The general approach is to use Convolutional Neural Networks (CNNs) to extract image features and Recurrent Neural Networks (RNNs) to generate tokens based on those features. This approach can be used to generate quaternion sequences from DRM images.

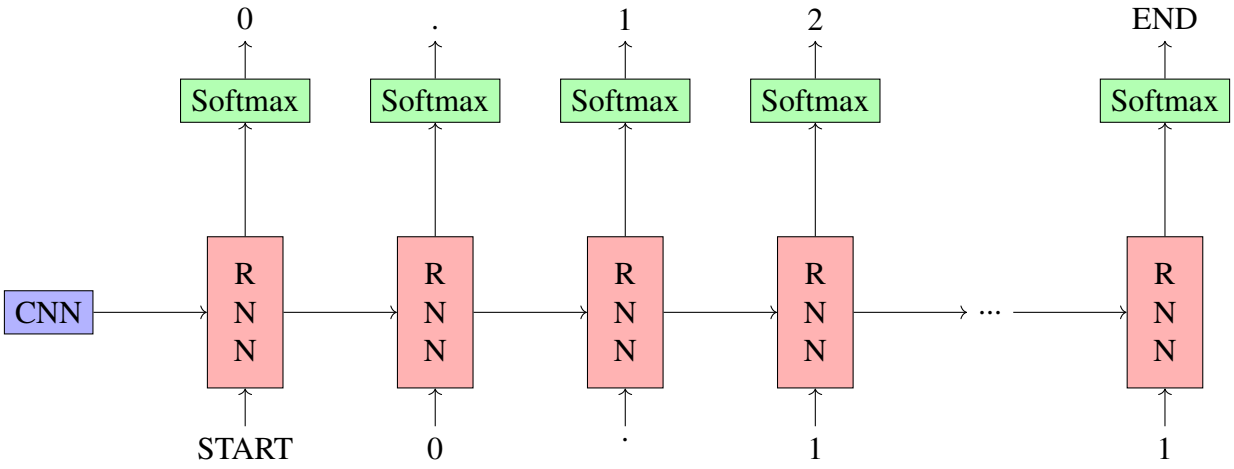

Figure 4. Training CNN and RNN flow Diagram. For the sequence, "0.123,-0.679,0.111,-0.321"

8.1 Feature Extraction with CNN:

Given an image I, we use a CNN to extract a fixed-size feature vector:

$$F(I) = \text{CNN}(I)$$

Where $F(I)$ represents the feature vector of image I. This feature vector contains information about the image's content and is extracted from one of the higher layers of the CNN, such as the fully connected layer before the final output layer.

In the diagram above, this extraction process is depicted by the 'CNN' box, which transforms the input image into a feature vector.

8.2 Sequence Generation with RNN:

Once the feature vector $F(I)$ is extracted, it is used as the initial state for the RNN. The RNN is trained to generate a sequence of tokens based on this feature vector. In the quaternion sequence, each letter in the quaternion string "0.123,-0.679,0.111,-0.321" represents a token. The number of time steps is length of this sequence.

The RNN operates in time steps. At each time step t, the RNN receives an input token w_{t-1} (from the previous time step) and produces an output token w_t. The RNN also maintains a hidden state h_t that helps it remember context across time steps.

The operations inside the RNN cell at each time step can be summarized with the following equations:

$$h_t = \sigma(W_h \cdot h_{t-1} + W_x \cdot x_t + b_h)$$
$$y_t = Softmax(W_y \cdot h_t + b_y)$$

Where:

- h_t is the hidden state's output.
- W_h, W_x, W_y are the RNN's weights.
- b_h, b_y are the biases.
- σ is an activation function, typically the softmax function.

The LSTM (Long Short-Term Memory) network is a modified and advanced type of RNN designed to handle long-term dependencies in sequences. While the basic structure is same as RNN, for longer ones or those requiring memory of past events much earlier in the sequence, an LSTM is preferred.

8.3 Softmax and Probabilities:

The Softmax function is used to normalize the output of the RNN for each word in the vocabulary, converting raw prediction scores for each word into probabilities.

Given a raw score vector z for all words in the vocabulary, the Softmax function is defined as:

$$\text{Softmax}(z_i) = \frac{e^{z_i}}{\sum_j e^{z_j}}$$

Where z_i is the raw score (also called logits) for word i, and the denominator sums over all words in the vocabulary.

The result is a probability distribution over all words in the vocabulary, where the probability of each word indicates how likely it is to be the next word in the sequence, given the current context.

8.4 Cross-Entropy Loss:

Cross-entropy loss is a commonly used loss function for classification tasks, including the sequence generation task in converting image to tokens.

Given a predicted probability distribution P and the true distribution T (usually a one-hot encoded vector representing the ground truth word), the cross-entropy loss L is defined as:

$$L = -\sum_i T_i \log(P_i)$$

Where the sum is over all words in the vocabulary.

The cross-entropy loss measures the dissimilarity between the true distribution (ground truth) and the predicted distribution. A lower value of cross-entropy indicates that the predictions are closer to the true values.

During training, the goal is to minimize the cross-entropy loss, which will result in the LSTM generating captions that are closer to the ground truth captions.

8.5 Training the Model:

For training, the model uses a reference caption (ground truth) for each image. The objective is to minimize the difference between the predicted sequence of words and the reference caption. This is usually done using a loss function like cross-entropy.

In the given diagram, the sequence "0.123,-0.679,0.111,-0.321" is transformed through a series of RNNs. Each RNN unit receives an input (e.g., "START", "0", ".", "1") and produces an output that is fed into a Softmax layer, which then predicts the next word in the sequence.

In summary, a CNN-RNN-based image captioning system uses the CNN to convert an image into a meaningful feature vector and then uses an RNN to transform this feature into a coherent sequence of words that describe the image's content.

8.6 Sampling from RNN Outputs:

Once the RNN produces a probability distribution over the vocabulary using the Softmax function, the next word in the generated caption can be chosen in various ways. One common method is sampling.

Sampling means choosing a word from the vocabulary based on the predicted probabilities. This introduces randomness in the generation process, which can be useful for producing diverse sequences.

Given a probability distribution P produced by the Softmax layer, the probability of selecting word i is P_i.

Greedy Sampling:

The simplest form of sampling is the greedy approach, where the word with the highest probability is chosen:

$$w_t = \text{argmax}_i P_i$$

Random Sampling:

A more random approach is to sample a word from the distribution P directly. In this method, each word is chosen based on its probability P_i.

The choice of sampling strategy can influence the quality and diversity of the generated captions. While greedy sampling often produces grammatically correct captions, they may be generic. More randomized sampling methods can produce more diverse and sometimes more creative captions, but with a potential trade-off in coherency or accuracy. For our purpose of generating valid quaternions, we can take a probability cutoff or cumulative probability cutoff to get valid output sequences.