Abstract Visual Reasoning by Finetuning Pretrained Transformers

Vedang Waradpande
Rutgers University
Department of Computer Science
vw120@scarletmail.rutgers.edu

Krishna Swaroop Pamidimukkala
Rutgers University
Department of Computer Science
kp956@scarletmail.rutgers.edu

Advait Bhat
Rutgers University
Department of Computer Science
ab2253@scarletmail.rutgers.edu

Sarthak Sharma
Rutgers University
Department of Computer Science
ss3849@scarletmail.rutgers.edu

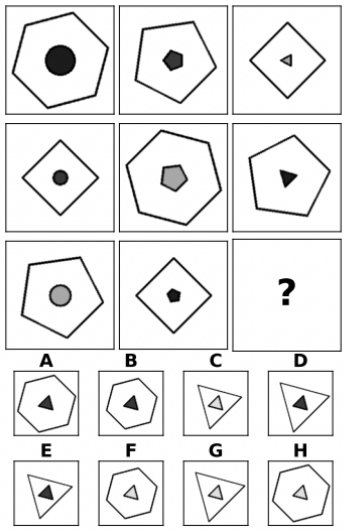
Abstract

Reasoning between entities and their relationships in a visual scene is a core requirement for Artificial Intelligence. Recently, several models were proposed and tested on the Raven's Progressive Matrices test to understand their ability to visually reason. Most Deep Learning models such as CNNs, LSTMs, and ResNets fail to perform close to human levels, highlighting the need to develop models that perform better reasoning with visual data. Transformers, with their multi-head self attention mechanism allow every input token to attend over every other input token, creating a natural framework for reasoning. In addition, the recent development of large, pretrained image transformers such as Vision Transformers (ViT) allow us to leverage their understanding of images while training them for reasoning tasks. We propose fine-tuning of two pretrained models - BEiT and ViT with an MLP head and Scattering Compositional Learner head respectively for Abstract Visual Reasoning. We test our models on the I-RAVEN dataset and compare them to previously tested baselines. To the best of our knowledge, these are the first pretrained transformer-based reasoning modules for solving RPMs.

1. Introduction

Abstract Visual Reasoning is the ability to analyze information, discover rules at an intangible level, and solve problems in innovative ways. The Raven's progressive matrices (RPM) [1] test is an IQ test used to examine and is highly correlated to a person's ability of abstract visual reasoning. The RPM consists of 3×3 matrix, with 8 context panels and 8 choice panels.The goal is to choose the correct choice panel to infill in the final slot of the RPM matrix. A sample RPM problem is illustrated in Fig 1.

Figure 1. RPM Matrix example



This test is thus ideal for understanding if Machine Learning models can perform end-to-end abstract visual reasoning. Until the last decade, most models focused on efficient representation learning of images and performing well on other Computer Vision tasks. Deep neural networks like Convolutional Neural Networks (CNNs) [2] and Long Short-term Memory Networks (LSTMs) [3] have demonstrated (sometimes superhuman) ability in a variety of vision and language domains. However, early tests with RPMs revealed that these algorithms do not reach near human capabilities at reasoning with visual cues.

Over the last few years, several models have been proposed to improve the performance of Deep Neural Networks on Abstract Visual Reasoning, where inductive biases have been used to propose efficient reasoning modules. However, the transformer architecture [4] provides a natural framework for reasoning, since the multi-head self attention allows each input token to attend to every other input token.

Recently, there has also been a focus on pretraining large transformers to learn the compositional structure of images or language (as a language model). These models have been extensively used in the field of NLP, and finetuning them has been known to be much more advantageous as compared to training new models from scratch. They provide improvements in terms of metrics such as accuracy and smooth and fast convergence.

In our work, we aim to leverage these pretrained models for abstract visual reasoning. Namely, we have the following contributions:

- We present some of the first pretrained transformers for abstract visual reasoning and test them on the I-RAVEN dataset.
- We compare these models to simpler Deep Learning models as baselines.

Our report is organized as follows: Section 2 explains related work in the field of Computer Vision and Abstract Visual Reasoning. Section 3 explains the models we propose and Section 4 explains our experiments, baselines, and results. Section 5 concludes our findings.

2. Related Work

Transformers in Vision. Based on the success of self-attention based architectures such as the Transformer [4] in Natural Language Processing, [5] introduced a simple extension of the transformer to Computer Vision tasks such as image classification named Vision Transformer (ViT). The image $\mathbf{I} \in \mathbb{R}^{H \times W \times C}$ is first broken down into N patches of size $T \times T$, where $N = HW/T^2$. These patches are flattened and linearly embedded before being passed through the transformer encoder. The encoder blocks consist of Multi-Head Self-Attention and MLP blocks interleaved with normalization layers. The output of the transformer, $g(\mathbf{I})$ consists of $N + 1$ tokens, the first of which is the output corresponding to the class token (`[CLS]`), which is fed to an MLP for classification tasks. The ViT model is pre-trained with a classification-like objective on the ImageNet dataset [6].

Pre-training transformers in a self-supervised fashion by corrupting data has been proven to be effective in NLP tasks. Notably, the BERT model [7] has been trained using a Masked Language Modelling objective, and BART is a sequence-to-sequence model trained using various types of corrupted data [8]. Inspired by BERT, the BEiT model [9] introduces a similar "Masked Image Modelling" objective for pre-training a ViT. The BEiT model claims to outperform other variants of ViT on downstream tasks like image classification and semantic segmentation and also improves convergence speed and stability during fine-tuning.

Abstract Visual Reasoning on RPMs. Early computational models for solving RPMs focused on the symbolic representations of the image panels, but recent work has included the visual aspect as well. This is inspired by the success of Deep Learning models such as LSTM [3], CNN [2] and ResNet [10] on Computer Vision tasks. The introduction of the PGM [11] and RAVEN [12] datasets further accelerated research in this area. Hu et al. [13] recognized a defect in the RAVEN dataset; for a given RPM, the correct answer can be identified without looking at the context panels by selecting the choice panel with the most common values for each attribute in the choice panels. To resolve this, they introduced the I-RAVEN dataset which uses an Attribute Bisection Tree to produce the choice panels. We use this dataset for our experiments.

Several models have been proposed over the years to solve RPMs visually. Barrett et al. propose the Wild Relational Network (WReN) [11] which applies the Relational Network module [14] multiple times to discover inter-panel relationships. The Dynamic Residual Tree (DRT)-based reasoning module proposed in [12] learns the structure of the underling symbolic language of the RPM and can be used in conjunction with any efficient feature extraction models. The work that followed such as CoPINet [15] and MXGNet [16] leveraged relations inside each row and column to create a classification score for each choice panel. SRAN [13] learns rule embeddings on different levels uses a gated fusion module to aggregate this information. The SCL model [17] uses three types of multi-head neural networks: the object, attribute and relational networks to execute a "scattering transformation" (we discuss this model in detail in Section 3).

Recently, there has been some effort to use a transformer-based architecture as a reasoning module for solving RPMs computationally. The ARNe model [18] used a WReN-like CNN to extract image features which were then used to train a 6-layer transformer for reasoning. An and Cho [19] introduced a hierarchical transformer encoder for reasoning. In this architecture, a 2-layer entity transformer first creates panel-level embeddings which are utilized by a 4-layer panel transformer to extract panel-level relationships.

3. Models

We present the architecture of the two transformer-based architectures that we propose.

3.1. BEiT-MLP

We fine-tune a pretrained BEiT encoder for image classification.

Pretraining. The model is pretrained as mentioned in [9]. As in a standard ViT model, the image of shape $3 \times 224 \times 224$ is divided into patches of shape 16×16 and a linear projection layer is used to create embeddings of these patches (we refer to these as "tokens"). 40% of the tokens are masked randomly to corrupt this image. The pre-training objective is to maximize the log-likelihood of the correct visual tokens z_i given the corrupted image.

$$\mathcal{L} = max \sum_{x \in \mathcal{D}} \mathbb{E}_{\mathcal{M}} \left[\sum_{i \in \mathcal{M}} \log p_m(z_i | x^{\mathcal{M}}) \right] \quad (1)$$

Where $x^{\mathcal{M}}$ is the corrupted image and $\mathcal{M}$ refers to the masked positions in the image. We use the pretrained checkpoint from [9] available in the HuggingFace API [20].

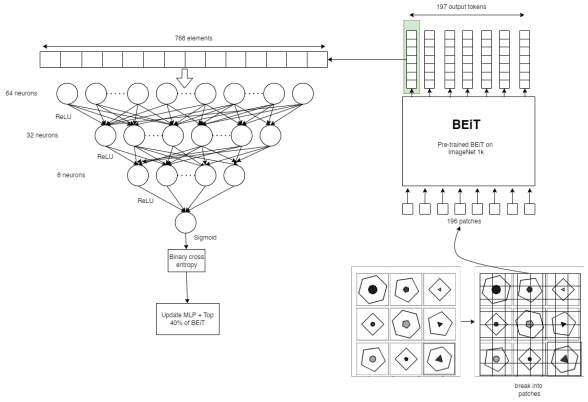
Fine-Tuning. The forward pass is presented in Fig 2. The input is a complete RPM image (with a candidate filled into the final slot). This is then divided into 196 patches of shape 16×16 and given as input to the BEiT model. We pass the output of the `[CLS]` token of the pretrained BEiT model to a 4-layer MLP with 64, 32, 8 and 1 neurons in its layers respectively. The fully connected layers are interleaved with ReLU activation functions [21] except the final layer neuron, which uses a Sigmoid $\sigma(.)$ activation function.

$$\sigma(x) = \frac{1}{1 + e^{-x}} \quad (2)$$

While training, we create 8 copies of the input with each candidate filled into the final slot in each of the copies. One of these 8 copies is a correct sample ($y = 1$) for the model, while the other samples are negatives ($y = 0$). We optimize the final 40% of the parameters of the BEiT model and all trainable parameters of the MLP using a binary cross entropy loss over these binary labels.

$$\mathcal{L} = BCE(\widehat{y}_i, y_i), \text{ where } \widehat{y}_i = f_{mlp}(f_{beit}(\mathbf{x}_i)[CLS]) \quad (3)$$

Figure 2. Forward pass of the BEiT model.



Augmentation. To reduce the significant class imbalance caused by converting the loss to a binary classification problem, we augment the positive samples with different transforms - rotate by $90°$, rotate by $-90°$, rotate by $180°$, horizontal mirroring, vertical mirroring, and switching the first two rows.

Inference. During the testing phase, we create 8 copies (one with each candidate infilled) of the RPM and calculate the final layer logits of each of the samples. We predict the sample wih the highest logit as the predicted correct choice (ranked prediction).

3.2. ViT-SCL

We train a Scattering Compositional Learner (SCL) [17] which accepts inputs from a pretrained Vision Transformer. The ViT is not fine-tuned during the training process, since its role is to extract efficient image features, not as a reasoning module. The roles of ViT here is to provide the SCL with deeply bidirectional embeddings of the RPMs.

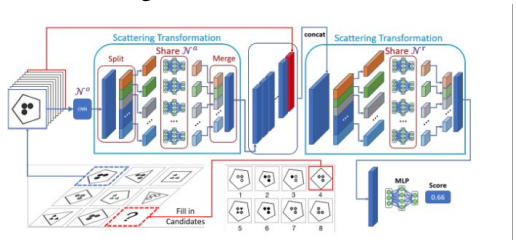
Pretraining. We use a pretrained checkpoint of the ViT-base model pretrained on ImageNet 1K [5] available in the HuggingFace API [20]. The SCL is trained from scratch.

Architecture. The SCL model uses three types of neural networks - object networks, attribute networks, and relationship networks in order to discover rich compositional structures of the data. This involves making eight copies of the 3×3 matrix with a different candidate in-filled in the final patch and passing these through the object network to extract object-level embeddings for each of them. The scattering transformation splits the representation first along the row-axis and then along the column-axis and feeds them through the attribute and relational networks respectively. [17] provides a deeper explanation of the architecture.

Fine-Tuning. We use the ViT encoder to generate embeddings for all 16 context and choice panels, which are then given as input to the SCL model. The SCL expects as input question and answer panels, which are reconstructed from the `[CLS]` token using a reconstruction layer, $f_r(.)$.

$$\mathbf{I} = f_r(\mathbf{W}\mathbf{X}_{\mathbf{cls}} + \mathbf{b}) \tag{4}$$

Figure 3. SCL architecture



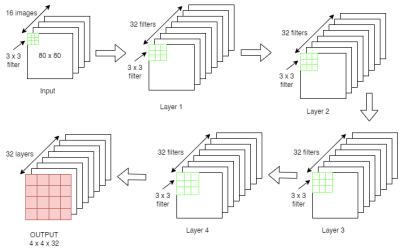
4. Experiments

4.1. Baselines

We explain the models we used as baselines for comparison for our model.

CNN-MLP. The CNN is interleaved with batch normalization and ReLU nonlinearity. Each layer has 32 learnable filters; this value was selected after rigorous experimentation. Increasing the number of filters causes the model to overfit much sooner. We use the standard 3×3 kernel and a stride of 2. The output of the final Convolutional layer is flattened before feeding it to the MLP. The two fully connected layers of the MLP have 512 neurons each, and the output is fed to a 8-dimensional softmax layer for the classification problem. Random dropout is applied at the penultimate layer of MLP. The input is a concatenation of all context and choice panels (16 images in total, each of size 80×80). The model is trained using a cross-entropy loss and the Adam optimizer. The architecture of this model is presented in Fig 4.

Figure 4. CNN architecture



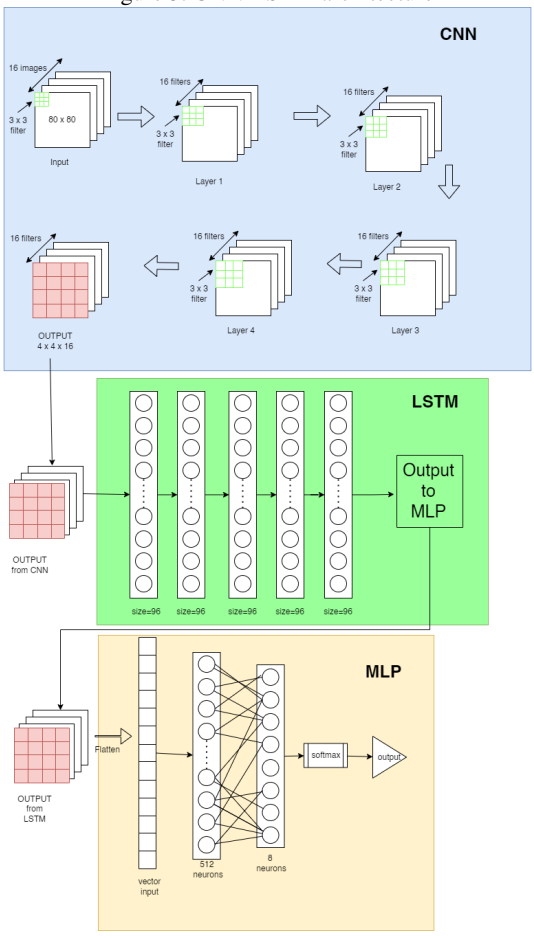
ResNet. We train a ResNet18 model [10] from scratch. The ResNet18 model is built with 18 convolutional layers of various kernel sizes with residual connections. The losses and optimizer are the same as the CNN-MLP.

CNN-LSTM. The CNN is interleaved with batch normalization and ReLU nonlinearity. Each layer has 16 learnable filters; this value was selected after rigorous experimentation. Increasing the number of filters causes the model to overfit much sooner. We use the standard 3×3 kernel and a stride of 2. The output of the final Convolutional layer now fed into an LSTM model with 5 layers with a hidden size of 96. Similar to CNN, based on the results of our experiments, we have picked the number of layers of LSTM carefully to avoid overfitting. The output from the LSTM layers is then fed into a 2-fully connected layers which eventually is fed into an MLP.

The two fully connected layers of the MLP have 512 neurons each, and the output is fed to a 8-dimensional softmax layer for the classification problem. Random dropout is applied at the penultimate layer of MLP.

The input is a concatenation of all context and choice panels (16 images in total, each of size 80×80). The model is trained using a categorical cross-entropy loss and the Adam optimizer.

Figure 5. CNN-LSTM architecture



WReN. WReN is a model which we apply Relation Network multiple times. This is done in order to understand the relationship amongst the various panels and to make a decision based on it. The WReN model works on Relational Network, the architecture of which has been displayed below.

The Relation Networks take multiple choices and compare it with a given option to check which catergory it belongs to. The Relation networks consist of 2 parts - f_φ and g_ϕ.
$g_\phi$is the embedding module where the features of the context panels and the choice panels are calculated. The f_φ is the relation module where we concatenate the feature maps of the context maps and output choice panels.

In our WReN, g_ϕ calculates the pair-wise similarity between the pair of context and choice panel. Then f_φ find a final score for each output choice panel. This is then passed through a softmax function and then we find the final answer.The WReN provides a score for each canditate panel, independent of the other choice panels.

4.2. Dataset

The I-RAVEN dataset is improved version of the RAVEN dataset. It was found after various experiments that the RAVEN dataset unexpected pattern among the eight multiple-choice panels as each answer panel is generated by randomly modifying one attribute of the correct answer. To overcome this drawback, the I-RAVEN RPMs are genrated using Attribute Bisection Tree algorithm to generate an impartial answer set for any attribute-based RPM question. The ABT ensures attribute modifications among the answer set are well balanced.

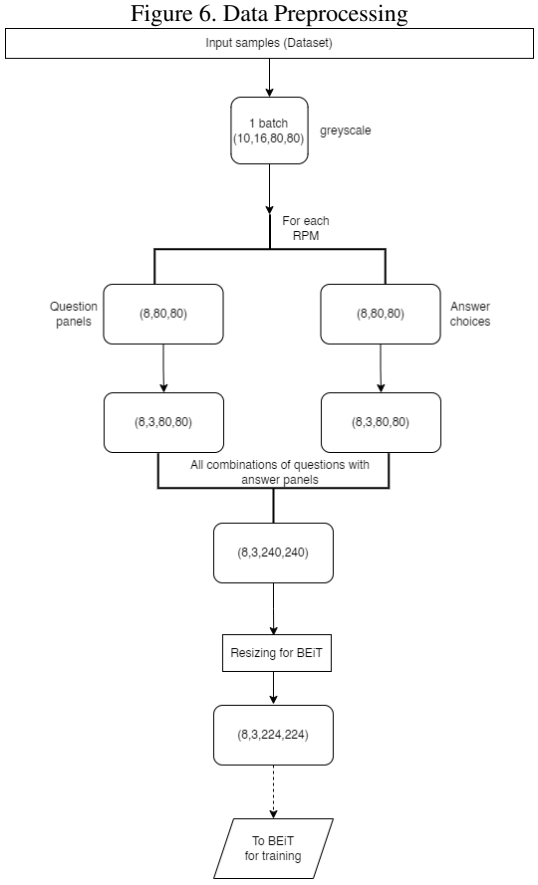
We use the I-RAVEN dataset , which is constructed using attributed Stochastic Image Grammar to evaluate our models. We use Center Single, 2×2 grid, Left-Right, and Out-In Grid samples.

4.3. Preprocessing

Before training the CNN-LSTM, we resize our context and choice panels to 80×80 from their original size of 160×160. Before concatenating the choice panels with the context panels, the choice panels are shuffled for each sample.

The pretrained BEiT requires a specific image size, for which we need to convert our input sample accordingly. We take a batch of 6 RPMs for training. Each RPM has 16 images (8 questions 8 answer choices). For each RPM in the batch, we first separate the questions panels and answers panels. Then for each of the context and choice panels we convert them from grayscale to 3 channels by repeating the same image. At this stage, to reduce class imbalance, we introduce the augmentations as mentioned in Section 3. Then we combine each answer panel with the question panels to give us 8 complete RPM images. We then resize the batch to a shape of $6 \times 3 \times 224 \times 224$, which can be given as input to the transformer.

Figure 6. Data Preprocessing



4.4. Setup

For all of the models, the parameters of the Adam optimizer are as follows: $\beta_1 = 0.9, \beta_2 = 0.99, \epsilon = 10^{-8}$ and the learning rate is 0.0001. We also experimented with the SGD optimizer for finetuning the BEiT model. The batch size is set to 64 for the baseline models and 6 for the transformer-based models because of memory limitations. We fine-tune the ViT for upto 100 epochs with early stopping on the validation set, and 10 epochs for the BEiT model. We use PyTorch for implementation and we use the hugging face API to obtain a pretrained version of BEiT and ViT. Models were trained on both Rutgers iLab (NVIDIA RTX A4000) and Amarel (NVIDIA GeForce RTX 2080Ti) clusters.

4.5. Results

With the settings discussed in the above sections, the ResNet model is able to outperform all the other baseline models i.e CNN-MLP, CNN-LSTM and WReN.

The ResNet achieved a maximum validation accuracy of

Model	Epoch with Max Val. Acc.	Max Val. Acc.	Test Acc. at Epoch	Center Single	2 x 2 Grid	Left-Right	Out-In Grid
CNN-MLP	15	23.96	24.37	20.54	20.18	22.73	22.56
ResNet	96	43.55	44.53	40.03	31.94	46.18	47.57
CNN-LSTM	51	16.32	15.74	15.36	15.86	15.24	15.16
WReN	19	23.82	22.45	23.65	21.5	17.25	18.83
SCL (Baseline)	45	77.54	78.87	83.90	70.62	79.85	82.35
ViTSCL	75	72.23	72.04	93.60	72.76	53.02	66.71
BEiT	32	25.33	26.87	23.66	21.98	21.41	26.11

Table 1. Results for the CNN-MLP, ResNet, LSTM, WReN, ViTSCL and BEiT models.

43.55%. We have compared the results obtained from all the model in Table 1. We have also presented the results when the models were tested on separate configurations: Center Single, 2×2 Grid, Left-Right, and Out-In Grid.

We observe that in case of CNN-MLP, Resnet models, they perform well on Left-Right and Out-In Grid configurations as compared to the other two configurations.

However, On the contrary, it is observed ViTSCL and WReN models are able solve the center-single RPM with greater accuracies as compared to the other configurations.

At a configuration level, we observe that the proposed ViTSCL model outperforms the baseline SCL model in center single and distribute four configurations but falls short in the rest of the configurations.

Results of BEiT: After training the BEiT model for 50 epochs, we observe that this model is able to outperform the CNN-MLP model but falls short as compared to rest of the baseline models.

The transformer architecture although intends to extract deeply contextual embeddings for the RPM and choice panels, is unable to perform the reasoning aspect of solving the RPMs.

Results of ViTSCL: With respect to ViTSCL, we observe that the model clearly outperforms the 4 initial models both at an overall level as well as for each of the configurations.

After running the ViTSCL model for 150 epochs we found out that the model was overfitting around the 75th epoch. Our best validation accuracy was around 72.23

At a configuration level, we observe that the proposed ViTSCL model outperforms the baseline SCL model in center single and distribute four configurations but falls short in the rest of the configurations.

Our model was not able to surpass the SCL model. This can be due to two major factors:

- **Overfitting of the model during training:** This can be the result of the complicated architecture of the model
- **Using non-Finetuned ViT:** We know that the ViT is pretrained on Imagenet1K dataset. Since, we have not finetuned the Vision transformer in order to make it better suitable for our tasks

We observed that the models learn well when we shuffle the target panels. We also compared the `shuffle=False` vs the `shuffle=True` settings in Table 2. It was done only for the CNN-LSTM model.

Metric	Shuffled	Not Shuffled
Train acc.	28.867	42.856
Val acc.	16.323	13.991
Test acc.	15.742	13.212

Table 2. Results for the LSTM model with shuffled and not shuffled data

Metric	Shuffled	Not Shuffled
Train acc.	69.10	63.66
Val acc.	23.96	12.92
Test acc.	24.37	12.91

Table 3. Results for the CNN MLP model with shuffled and not shuffled data

Ablation Study. We test how the performance of the models changes with size of the training set. For each model, we start with 20% of the training data (chosen randomly) and keep increasing the proportion by 20% in each step.

The Fig. 7 shows the comparison of how the 4 models performed on different percentage of training data. We can see a general trend on how the performance increases as more data is provided. We see significant improvement in the performance of the ResNet model as we give the model more training data.

Performance with Ground Truth Factors. Given the different factors that go into creating the RPM, it is critical that we look at the performance of each individual aspect.

We've created data with different color, size, and form variables. We have observed that all the baseline models have consistently performed well when color attribute is frozen and the model was exposed to data with variation created by varying the other attributes.

Model	Color	Size	Type
CNN-MLP	28.74%	21.89%	19.48%
Resnet	45.67%	28.12%	20.82%
CNN-LSTM	41.21%	19.10%	18.54%
WReN	43.29%	40.34%	38.76%
ViTSCL	71.97%	7.67%	58.61%
BEiT	32.47%	30.65%	26.13%

Table 4. Performance of models on Ground Truth Factors

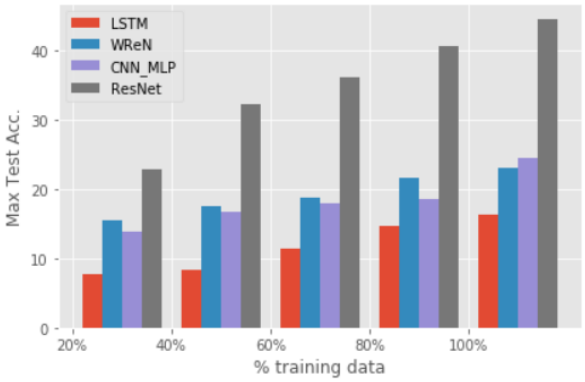


Figure 7. Test accuracy of all models by percentage of training set used.

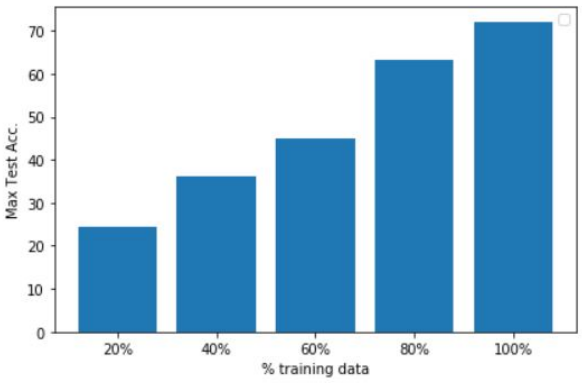


Figure 8. Test accuracy of ViT SCL by percentage of training set used.

Observations from Training, Validation and Testing loss and Accuracy Plots We have also plotted graphs to show the training, testing and validation accuracy trends for all the models.

In figures below we show the performance vs training iterations. We can see that after few epochs the model is overfitting on the training data. The test and validation accuracy stabilizes.

Some interesting trends that we observe are: (1) ResNet model generalizes well than other models (Fig 10). It overfits after the 50th epoch. (2) For the Vit + SCL (Fig 13) we trained the model for 150 iterations. We saw that the model was overfitting around the 75th epoch, the validation loss starts increasing.

In general, we observe that the loss values in both the training and validation phases is constantly decreasing for all the models.

5. Conclusion

We propose pretrained transformer-based models for Abstract Visual Reasoning on the I-RAVEN dataset. While the BEiT-MLP model can learn on the training dataset well, it cannot generalize well to unseen data. Data augmentation and minimizing class imbalance were crucial for training the BEiT-MLP. When the transformer is used for extracting image features in conjunction with a good reasoning module (ViT-SCL), we achieve good performance on the test set. However, this model still falls short of the vanilla SCL model.

Furthermore amongst the baseline models, we saw that ResNet18 model outperforms all the other baseline models but we do observe some variation in performance of the model over different configurations.

We hope our work provides a framework for further research into Abstract Visual Reasoning with pretrained transformers. There are simple extensions of our work which can have some promise:

- Masked pre-training of BEiT in a leave-one-out fashion can help the transformer understand the rules in a more robust manner.
- Better data augmentations to reduce class imbalance.

6. Project Code

The code for building and training all the proposed and baseline models can be found at https://github.com/VedangW/AbstractVisualReasoning

References

[1] JC ea Raven. Raven's progressive matrices. *Western Psychological Services*, 2:5, 1938. 1

[2] Yann LeCun, Bernhard Boser, John Denker, Donnie Henderson, R. Howard, Wayne Hubbard, and Lawrence Jackel. Handwritten digit recognition with a back-propagation network. In D. Touretzky, editor, *Advances in Neural Information Processing Systems*, volume 2. Morgan-Kaufmann, 1989. 1, 2

[3] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997. 1, 2

[4] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. 2

[5] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020. 2, 3

[6] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009. 2

[7] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018. 2

[8] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *arXiv preprint arXiv:1910.13461*, 2019. 2

[9] Hangbo Bao, Li Dong, and Furu Wei. Beit: Bert pre-training of image transformers. *arXiv preprint arXiv:2106.08254*, 2021. 2, 3

[10] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 2, 4

[11] David Barrett, Felix Hill, Adam Santoro, Ari Morcos, and Timothy Lillicrap. Measuring abstract reasoning in neural networks. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 511–520. PMLR, 10–15 Jul 2018. 2

[12] Chi Zhang, Feng Gao, Baoxiong Jia, Yixin Zhu, and Song-Chun Zhu. Raven: A dataset for relational and analogical visual reasoning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5317–5327, 2019. 2

[13] Sheng Hu, Yuqing Ma, Xianglong Liu, Yanlu Wei, and Shihao Bai. Stratified rule-aware network for abstract visual reasoning. *arXiv preprint arXiv:2002.06838*, 2020. 2

[14] Adam Santoro, David Raposo, David G Barrett, Mateusz Malinowski, Razvan Pascanu, Peter Battaglia,

and Timothy Lillicrap. A simple neural network module for relational reasoning. *Advances in neural information processing systems*, 30, 2017. 2

[15] Chi Zhang, Baoxiong Jia, Feng Gao, Yixin Zhu, HongJing Lu, and Song-Chun Zhu. Learning perceptual inference by contrasting. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. 2

[16] Duo Wang, Mateja Jamnik, and Pietro Lio. Abstract diagrammatic reasoning with multiplex graph networks. *arXiv preprint arXiv:2006.11197*, 2020. 2

[17] Yuhuai Wu, Honghua Dong, Roger Grosse, and Jimmy Ba. The scattering compositional learner: Discovering objects, attributes, relationships in analogical reasoning. *arXiv preprint arXiv:2007.04212*, 2020. 2, 3

[18] Lukas Hahne, Timo Lüddecke, Florentin Wörgötter, and David Kappel. Attention on abstract visual reasoning. *arXiv preprint arXiv:1911.05990*, 2019. 2

[19] Jinwon An and Sungzoon Cho. Hierarchical transformer encoder with structured representation for abstract reasoning. *IEEE Access*, 8:200229–200236, 2020. 2

[20] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. Huggingface's transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771*, 2019. 3

[21] Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *Icml*, 2010. 3

7. Appendix

7.1. Contributions

- Report and Powerpoint Presentation: Entire Team
- Milestone 1: Krishna, Vedang and Advait
- Milestone 2: Vedang, Krishna and Sarthak
- Milestone 3: Krishna
- BEiT-MLP: Vedang and Krishna
- Vit-SCL: Krishna and Vedang

7.2. Further Results

We present the train-val-test accuracy graphs referenced in this report below.

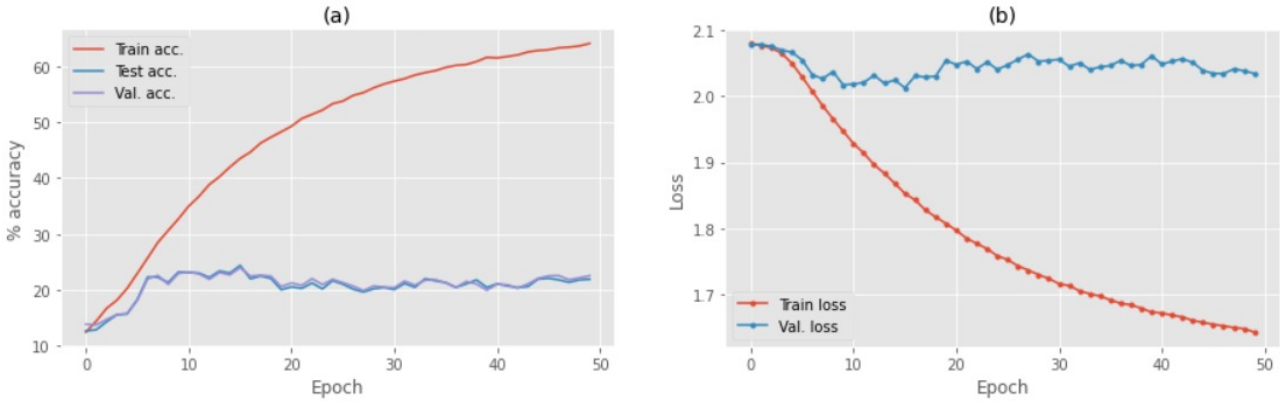


Figure 9. Result of training CNN-MLP for 50 epochs.. (a) Performance vs training iterations (b) Loss vs training iterations.

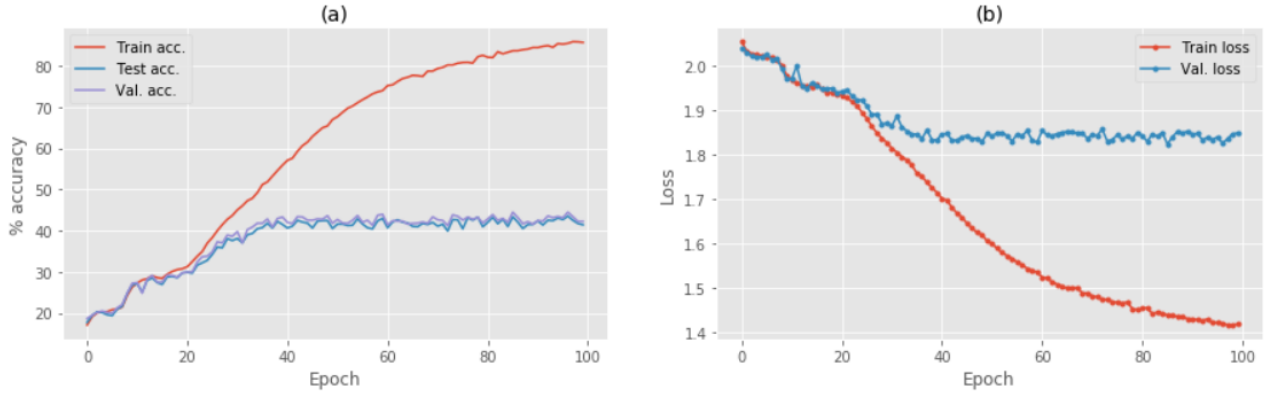


Figure 10. Results of training ResNet for 100 iterations. (a) Training accuracy increases quickly and reaches over 90%. (b) Model overfits after the 30th epoch as the training and validation losses diverge.

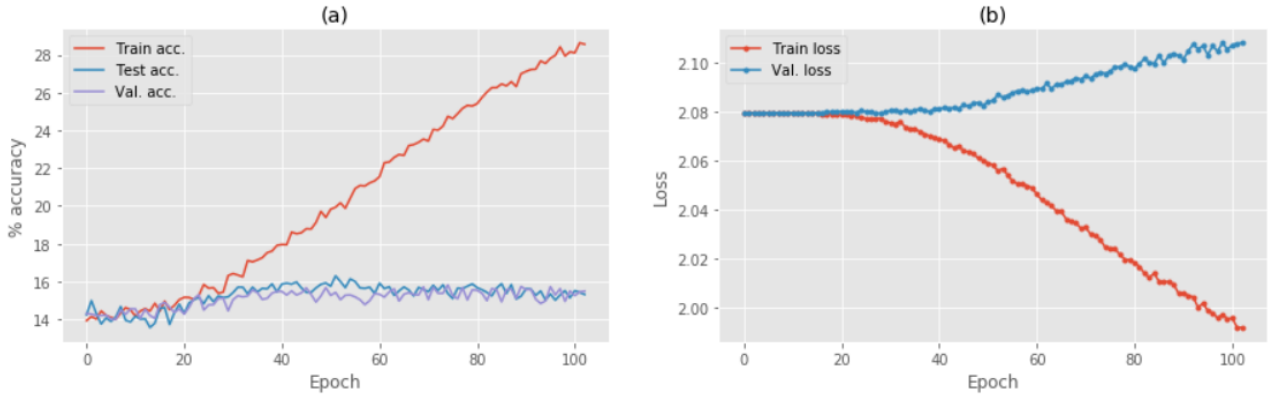


Figure 11. Results of training CNN-LSTM for 100 iterations. (a) Performance vs training iterations (b) Loss vs training iterations.

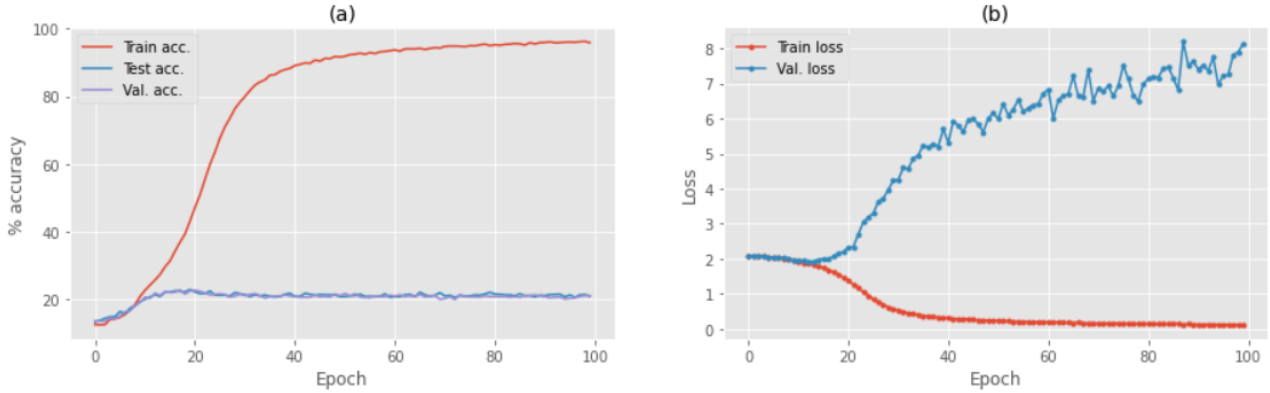


Figure 12. Results of training WReN for 100 iterations. (a) Performance vs training iterations (b) Loss vs training iterations.

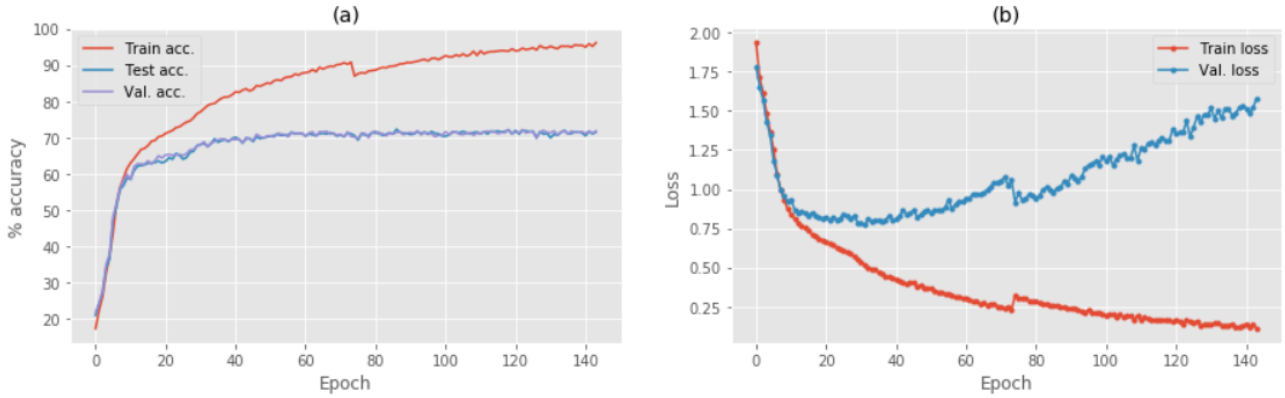


Figure 13. Results of training ViT + SCL model for 150 iterations.

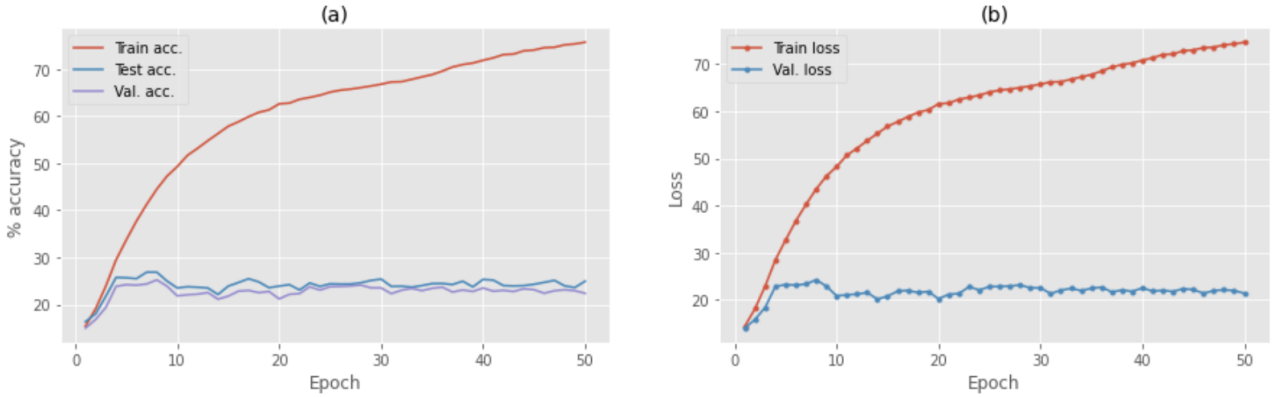


Figure 14. Results of training BEiT model for 50 iterations.