Volume: 13 Issue: VII Month of publication: July 2025
DOI: https://doi.org/10.22214/ijraset.2025.73281

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 13 Issue VII July 2025- Available at www.ijraset.com

Qonstraint 5.9: Efficient TinyImageNet Classification with a Novel Hybrid Vision Transformer and Classification Head

Naman Gupta
Undergraduate Student, Department of Computer Science, BITS Pilani KK Birla Goa Campus

Abstract: The paper talks about a model Qonstraint 5.9. This is an approach that takes into consideration many branches of efficient image classification on resource-constrained benchmarks, which is evident in the dataset used, namely TinyImagenet. This efficient approach doesn't commonly rely on optimization cheats like pretraining or aggressively resizing input data; instead, it is built from scratch to showcase the strengths of a convolutional-transformer hybrid model. The foundation of the same lies in QonvViT blocks, which are a beautiful crochet of convolutional token mixing inspired by mobileViT, adaptive feed-forward processing, and optimized downsampling. The model learns from both local and global features. This makes the model learn a broader picture of the context as well as focus on the fine-grained details of the image. All this is done keeping in mind resource constraints alongside issues that arise from the given condition of not using pretrained weights and layers. How this is achieved is explained in the sections to come. One of the many things that sets Qonstraint apart is not just making things dynamic, it's the way it makes parameters and behaviors dynamic. It is based on a machine-learning-inspired reinforcement learning approach, utilizing a confidence-awareness training regime. As you scan through the model, one feature that will be vividly noticed is the intensity regularization and feature scaling being modulated in real time, but still not causing an overhead to model efficiency. This makes the model allocate its compute to where and when it's needed on the fly. What follows is an optimized classifier head, which is a sophisticated ensemble of global, local, and auxiliary regularization, whose outputs are then again fused based on confidence. Qonstraint an adaptive solution that balances all these factors so as to address the issue of high-performance image classification in limited resource availability. The outcome is a model that not just acts as a head over renowned models, but becomes the foundation model itself, combining various SOTA techniques, setting a new standard for from-scratch learning in constrained environments.

I. INTRODUCTION

A. Motivation

Deep learning has gone through a flux of enhancements, which have brought vision transformers (ViTs) and hybrid CNN-ViT models to the driver's seat in image classification. Both of these approaches have achieved remarkable results on large-scale datasets, often leveraging pretraining and heavy computing, and resizing. Expectedly, this success doesn't flawlessly translate to the constraints posed by smaller benchmarks like TinyImagenet, CIFAR100, etc., where the data availability is very limited and the hardware limitations become more evident. When it comes to real-life scenarios, which range from embedded visions on smaller, mobile-like devices to research work with modest GPU domains, there is a visible need for models that can deliver without the tailwind of the comfort that comes from pretraining over large-scale datasets.
We have been witnessing an exponential increase in the number of hybrid models that attempt to blend convolutional and transformer processes, but most of the existing renowned solutions utilize heavy reliance on pretrained weights and resizing compute. These dependencies limit accessibility and sort of foreshadow the true strengths of the underlying architecture, thus resulting in the absence of achieving robust, ground-up learning on compact datasets, and the problem still being unsolved. The pressing demand for such motivates the founding of Qonstraint.

B. Problem Statement

The motivation mentioned above justifies how the gap of designing models with accuracy and practicality for small-scale databases, acknowledging real-world constraints. The majority of the SOTA architectures based on ViTs or ViT-CNN hybrids rely on external help, as justified. With Qonstraint, we aim to find a solution that doesn't rely on such factors, thus making it accessible to researchers and practitioners.

TinyImageNet is a very vague dataset with heavy granularity and limited samples, thus serving as a very challenging benchmark for evaluating the genuine adaptability of these SOTA architectures and Qonstraint aims to bring these together in a mosaic of the recent-most proved approaches into a framework viable for small-dataset image classification tasks, without any external supervision or resource abuse. This field lacks a solution that demonstrates high accuracy, efficient convergence, and memory efficiency, all whilst being trained from scratch and deployable on publicly available resource-limited GPUs. This model directly addresses this gap by proposing a framework that caters to these stringent requirements and becomes a new standard for efficiency and from-scratch image classification on these challenging compact datasets.

C. Contributions

The work that follows talks of the architecture named Qonstraint 5.9, a hybrid architecture that achieves and advances the state of efficient and ground-up image classification models on constrained environments. The model brings together a mosaic of numerous novel components and thought processes, each of which is designed to cater to a specific limitation of the existing architectures and thus can form a holistic adaptive learning system. The components that comprise the mosaic comprises of, which make Qonstraint breach the efficiency benchmarks, are as follows:

1) QonvViT Block: The custom building block that integrates convolutional layers with mobileViT-like token mixing, a hybrid architecture different from conventional hybrids, now having phases of hybridization but instead involving ViT tricks into the CNN framework, is named the QonvViT block with depth-wise feedforward networks. This design is brought together in a way that captures both a global context and a local structure refinement, while maintaining efficient computation and easy scalability options.
2) Confidence-based reinforcement training: The model works in an adaptive training environment in which feature scales, optimization hyper parameters, and even regularization intensities are modulated amidst the epochs running through based on recent and past behaviors, portrayed in the form of epoch’s confidence whilst ensuring stable convergence, it also possesses a mechanism which lets the model focus more on harder samples, and makes the further epochs focus more of their compute where it’s needed the most.
3) Multi-Path Classifier Head: Qonstraint is aided by a triple-path ensemble-like classifier head that gives dynamic weightage to three pathways simultaneously: a global reasoning transformer-based branch, a localized convolutional branch, and a regularizing auxiliary branch. The weightage of each head in the ensemble is confidence-based, which allows the model to adhere to the branches in an optimized ratio for every input sample.
4) Advanced Data Pipeline and Augmentation: The model utilizes a pipeline with strong and weak augmentation phases, which get toggled based on the epochs passed. Also applies batch-level augmentation, including CutMix and MixUp. These get applied via custom collate functions, which help add an extra layer of generalization without increasing model compute.
5) Resource-Efficient Design and Training: Qonstraint has been optimized and built for inference on the widely available GPU, specifically the Kaggle GPU, which even locally would require only a single mid-range GPU (e.g., NVIDIA P100, 10.2GB VRAM). The absence of any input resizing or pretraining demonstrates that strong performance can be achieved purely through novel architectural innovations.

The final model reported, being Qonstraint 5.9, is the 9th version of the 5th approach. The aftermath of 27 models before this one, utilizing light pretrained weights from various renowned datasets, model approaches, and thus making up for an elaborate ablation study. Qonstraint 5.9 thus demonstrates superior accuracy and convergence speed compared to conventional CNNs and ViTs in terms of accuracy efficiency.

II. RELATED WORKS

A. Efficient Vision Transformers

The drift from CNN networks to transformer networks brought about a surge of works. It began with the seminal transformer architecture for sequence modeling introduced by Vaswani et al. (1). Vision transformers adapted this approach to the domain of images by reshaping and representing images as a sequence of patches, which achieved state-of-the-art results on large-scale datasets in no time. Just the limitation being that it needed a large amount of compute, a very diversified and regulated large dataset, along with extensive pretraining (2). To overcome this limitation and smooth the drift, a variety of efficient and hybrid ViT variants have been proposed.
Tokens-to-Token ViT (T2T-ViT) talked of a progressive tokenization model that better preserved local image structures and enabled a really effective from-scratch training (3).

Then came DeiT, which became a large milestone in itself. It demonstrated that through strong data augmentation and knowledge distillation, a vision transformer can be trained from scratch on the full ImageNet, which narrowed down its gap to very developed CNNs (4). StructViT was another approach. It took advantage of a structured self-attention incorporation alongside a hierarchical design, which improved multiscale feature extraction and modeling whilst still maintaining computational efficiency (5). Another great, not-so-popular solution was SHViT. Single-Head Vision Transformer proved its ability to reduce redundancy by substituting multi-head attention with a single-head attention module and a parallel arrangement of global-local arrangement, thus significantly improving efficiency for resource-constrained devices (6).
Another approach, which was also tried out in Qonstraint 5.5, an iterative version of model 5.9 came up to, was the regime of neighborhood attention, which introduced an explicit attention mechanism of self-windowing. It provided a flexible and memory-efficient global self-attention (7). The same posed up to be the inspiration for Qonstraint's token mixer, having been updated in MobileViT. MobileViT was the pioneer of the integration of convolutional token mixing with transformer-based global reasoning, which achieved high accuracy with low latency, thus making it applicable for mobile and edge applications (8).
Our proposed model, Qonstraint 5.9, directly draws on most of these advances as listed above. Convolutional token mixing and hybrid stacking inspired by MobileViT (8) and SHViT (6), a modular backbone and fusion blocks sort of draw a similarity with the hierarchical principles of StructViT (5) and T2T-ViT (3). The hybrid confidence-driven regularization and ensemble classifier design are novel additions, but still draw insights from the broader literature on the domain (4, 12).

B. TinyImageNet Benchmarks

The dataset we are presenting the performance on, TinyImagenet, a subset of ImageNet, consists of over 100K images spanning 200 classes at a very minimalistic resolution of 64X64, thus justifying the challenges it poses. It is widely used as a benchmark to evaluate model efficiency and regularization powers under very constraining data and computational parameters. Well-known CNN architectures like ResNet and DenseNet have been very extensively optimized specifically for this dataset, but both of them plateau beneath 55% accuracy when trained on model architecture only without utilizing pretrained weights, but still with an optimized conventional weight initialization (9,10). The newer, more recent hybrid and transformer models include MobileViT and Deit-Ti, which have showcased that careful architecture and augmentations can help close the gap. The limitation still being that these depend on pretraining or external influence in the name of distillation (8,4). The ever-sustained challenge of achieving high accuracy values without any pretraining on TinyImageNet thus backs the need for this novel, resource-efficient architecture of Qonstraint 5.9.

C. From-Scratch Training

When talking of applications in domains where pretraining data is unavailable, or the domain of the data is mismatched, it becomes really important to train deep vision models from scratch without dependency on large-scale pretraining. DeiT was one of the first studies establishing a benchmark, which showed how transformers can match CNNs when trained from scratch, given effective mechanisms and augmentations (4). T2T-ViT's convergence and generalization without any external data were enabled by progressive token aggregation (3). Research in this domain is fairly recent and shows that maximizing training dynamics improves generalization by accelerating convergence and an adaptive implementation of regularization (12).

D. Limited resource architecture

All the models that have shown success in other domains have failed to be interpretable in applicable domains, the reason being they require excessive compute, which is not available on mobile and edge devices, thus limiting the deployment possibilities of these models. The same demands for efficient models that balance accuracy and computational resource utilization. MobileViT countered the same with inverted residual blocks. Combining the same with a separable attention mechanism made up for a milestone in this domain, reducing latency and memory footprints while not compromising performance to inefficient degrees (8). SHViT's stride-optimized patch embedding enhanced speed (8). Natten and adaptive token pruning pushed this frontier, having enabled real-time adjustment of resource usage during interference (7,13).

E. Novel Classifier Head

Qonstraint possesses a triple-path classifier. It portrays a novel version of recent advances in multi-branch and ensemble learning. Leveraging multiple pathways for apt decision making is something that has been explored before in medical image and partially supervised learning, where triple-view architectures have demonstrated improved generalization and stable variance trends (14,15).

Confidence-driven fusion of the same, being a novel territory, ensures that the most reliable pathway is given the privilege on each input, drawing similarities from recent work on adaptive ensemble methods.

III. METHOD

A. Overview and Pipeline Diagram

The architecture of Qonstraint 5.9 is optimized for performing end-to-end image segmentation tasks, particularly on low-resolution and more confined datasets, such as TinyImageNet. The model's pipeline is a novel hybrid architecture, taking advantage of ViT tricks in a CNN network. The pipeline begins with a stem of convolutional layers meant to extract low-level features and performs spatial downsampling from the get-go. This prepares the input for an efficient processing line coming up.

The output from this stem is followed by multiple stages of QonvViT blocks, which are a sophisticated collection of convolutional token mixing with a lightweight localized neighborhood attention. This design efficiently balances local spatial features and global context. This enables feature extraction with linear computational complexity concerning the input size.

The different stages, namely the high_feat and low_feat, produce distinct feature maps. These are combined using a fusion module that fuses information via residual gating and applies cross-scale attention over the two stages of feature maps. This fusion is meant to endorse the representational richness of the features by synthesizing very specific local details with higher-level semantic features. These fused features are passed to the multi-branch classifier head, which we have been talking about. The outputs of the three branches, the convolutional, transformer-based, and auxiliary heads, are dynamically combined.

The model is designed to give significant attention to reproducibility and efficiency. The pipeline contains train-validation splits and two phases of augmentations alongside batch-level MixUp and CutMix via a collate function. The training for the same is performed on a single GPU with deterministic seeding for everything to ensure reproducibility of results.

This adaptive design is what allows Qonstraint to achieve high-accuracy values without any pretraining or input resizing, making it applicable in constrained deployment scenarios with proof.

B. Architectural Components

1) Convolutional Stem

Qonstraint starts with the application of a convolutional stem in the initial stages of the network. This stem is responsible for making the model extract low-level features. It transforms the raw 64x64 RGB images received into a smaller but informative dense feature map, which is used for subsequent attention processing.

The stem in question consists of three convolutional blocks, each of which has a 3x3 convolutional layer with increasing channel width. Each layer is followed by batch normalization, active dropouts, and a nonlinear activation layer utilizing ReLU. The first two of the three layers employ a stride of 2, resulting in a progressive layer-wise reduction of spatial dimensions. Statistically, the inputted raw images get transitioned from 64x64 to 64x64 after the first block. This is meant to just learn the edges and basic spatial features of the surroundings. The second block does another level of the same convolution, adding into the learning of a bit more complex features. The third block that follows samples down the size 2x, letting the tokens act up.

The design includes depthwise separable convolutions. Splitting convolutions into depthwise and pointwise operations drastically reduces the parameter count and floating point operations without harming the ability to capture local spatial patterns. Progressively downsampling also helps in spatial reduction via strided convolutions. This lowers the computational overload and enhances the model's focus on representations that become increasingly ambiguous to classify. The batch normalization and dropouts follow each layer and help stabilize the feature distribution and counter the biggest issues with small datasets having many classes early on, overfitting. ReLU activations, being non-linear, help in maintaining gradient flow and defend against vanishing gradients everywhere, acknowledging the network depth.

The stem design caters to the need of preserving as much local information as possible in early stages, given the small image size and hairline class distinctions of TinyImageNet. The stem provides a strong foundation to the QonvViT blocks and fusion module that follow this stem, certainly maintaining the ability to capture both local and global details throughout the network. What results is a highly efficient and stable initial representation of the data, which is crucial for the model's performance given no help from pretraining.

2) QonvViT Blocks

These are the blocks that the backbone stacks up as described later in the paper. This block functions like the brain of Qonstraint 5.9 and is the main unit to perform the task of feature extraction for the model.

These blocks aim to bring together the strengths of local context, aided by the convolutional network intertwined with channel interaction, and an efficient structure having adaptive modular regularization in a lightweight structure.

a) Token Mixing: The first step in the stream is a hybrid token mixing mechanism. The structure begins with a 3x3 convolutional layer with kernel size 3, stride 1, and padding 1, which captures local patterns and textures. The same is followed by a GELU activation and a 1x1 convolution, implemented to mix channel information and bring a sense of non-linearity. This two-staged convolution sequence results in a representation which is sensitive to both spatial and channel-wise dependencies, providing for a strong inductive bias, crucial for small resolution images.

b) Layer Normalization and Residual Scaling: Before and after the token mixing does its job, the feature maps from the stem are passed through a LayerNorm alongside flattening for LayerNorm. This ensures a stable feature distribution and acknowledges the issue of an internal covariate shift, a commonly seen issue with deeper models. The block also contains residual branches. These residual branches are scaled by the parameters res_scale1 and res_scale2. Both these parameters are learnable and thus allow the model to dynamically control the impact of each transformation during training. This residual scaling is vital to fulfill the need of maintaining training stability by stabilizing gradients and not letting them either explode or vanish, both of which are problems the model gets increasingly prone to with depth.

c) Channel MLP with Depthwise Convolutions: Once token mixing has done its thing, the data flow meets a second section of transformation in the form of a channel MLP. This is realized as a pair of 1x1 convolutions separated by GELU activation. The first convolution is meant to expand the feature dimension up to a factor of mlp_ratio (taken 4.0). The second convolution projects it back to the original dimension it received. This extra layer of non-linearity enhances QonvViT's expressivity and allows it to model complex channel interactions.

d) Token Excite Module: Now, one of the main modules of Qonstraint as a whole, the Token Excite Module. It is inspired by the way MobileViT implemented token mixing and squeeze-and-excitation mechanisms. After the transformations have happened, the feature map is flattened into a single sequence of tokens, which is passed through a lightweight MLP, which is followed by a sigmoid gating function. This mixer adaptively recalibrates the importance of each token. This spatial location emphasis enables the block to express salient features and dynamically suppress noise. This adaptive gating becomes critical for robust learning, given the strength of the data augmentations in the second phase of augmentations and the label noise.

e) Adjustable DropPath: Every QonvViT block incorporates a stochastic depth regularization. Unlike the conventional, well-known approaches, the drop probability isn't fixed. Adding this extra sense of dynamic nature, the drop probability is dynamically modulated at runtime based on the current confidence of the model, which is derived from its recent validation accuracy trends. When the training is going through a phase of higher confidence values, the model challenges itself by regularizing more aggressively. On the other hand, whenever confidence values are low and the momentum lessens as a result of it, the model caters to its uncertainty by easing out the regularization and sustaining more information flow.

f) Distributional Clamping: Whenever the feature maps are subjected to any case of major transformation, they are then clamped to a fixed range (taken [-6,6]) to prevent any distributional drift causing numerical instabilities. As an attachment to this clamping, the intermediate features thus derived are readjusted to shift the mean towards a neater bound around 0. This bounded range stabilizes training and keeps a check on the activations, ensuring that they remain well-behaved even when aggressive augmentations are coming into play through the deep stacks.

The QonvViT Blocks are structured with the motivation to address the need to combine the locality and efficiency of convolutions with the expressivity and adaptivity of transformer-style attention, whilst keeping the notion of a lightweight from-scratch trainable module.

3) QonvViT (backbone)

The arrangement is a modular deep stack. These complex QonvViT blocks, when put into play, are stacked into stages. Each of these stages increases the channel dimension and reduces the spatial resolution via strided convolutions. The hierarchical arrangement lets the network progressively capture the abstractions and richer semantics of the features. The design ensures that the principles discussed above, the hybrid mixing, normalization, excitation, and regularization, are consistently getting into play throughout the model's depth. This is where all the dynamic drop probabilities are assigned and mean adjustments are made.

In the full QonvViT backbone, the QonvViT blocks, which comprise all of what was discussed above, are organized into three hierarchical stages. This backbone returns both the intermediate feature states, namely the low_feat and the high_feat. These multi-scale features are then sent to the fusion block.

It has a _make_stage function. This performs the task of building the layers that collectively form a stage in the backbone. It adds a stack of QonvViTBlock series, which happens after the initial convolution and GELU activation. The number of blocks in this series is given in the depth parameter for that stage. (depths= [3,5,6]). Each block in the stack is assigned a drop path regularization rate, ensuring deeper blocks are relatively strongly regularized. The result is a ModuleList that contains both the initial downsampling convolution and a set of QonvViT blocks. These together form a unit for hierarchical feature learning. The hierarchy is created by repetitive application of the same. This backbone is meant to enable efficient depth and width scaling, which then supports representation at multiple scales throughout the backbone.

4) *FusionBlockV2*

The FusionBlockV2 is engineered in a way that integrates multi-scale features from different depths of the network, thus endorsing the model's ability to capture both high-level semantics and finer spatial details. This fusion begins by projecting both the high-level and low-level feature maps to a common dimension using 1x1 convolutions. This reduction of dimension ensures that the features from different stages are compatible for the actions that follow and minimizes computational overhead. To enact this alignment, the high-level feature map is up-sampled to match the resolution of the lower-level feature map using interpolation of a bilinear nature, preserving the spatial structure necessary for the effectiveness of the fusion. After the projection and alignments, both stages of feature maps are flattened to go through an attention mechanism. The low and high-level feature maps act as the query and the key, respectively. This mechanism models the relationships between features at different scales explicitly. This lets the network emphasize complementary information and fill the gap between shallow and deep representations. The output from the attention module is shaped back to spatial format, whilst ensuring that the information is spatially coherent and maintains the context.
The attention included output is now subjected to the lower-level feature map along the channel dimension. The tensor post concatenation is passed through another 1x1 convolution that mixes across channels and reduces the dimensionality back to the original token dimension. This operation blends the features from both scales and acts as a learnable gating mechanism, which allows the network to adaptively weigh the contribution from each source. The fused feature map then goes through another projection head of 1x1 convolutions, batch normalization, GELU activation, and dropout, which introduce regularization.
One thing that can be noticed in FusionBlockV2 is the extensive use of dropouts in the process, which by themselves encourage the network to learn generalizable feature combinations. This enables the production of rich representations with efficient computational costs.

5) *Classifier Heads*

a) *SOTAQlassifierHead*

The classifier head that the model utilizes, namely the SOTAQlassifierHead, is a multi-stage multi-branch ensemble designed to aggregate and regularize the feature representations before the prediction happens. The design is motivated by the requirement of a balanced, globally reasoning, and locally aware classification head, specifically in the context of a small-scale but largely diverse classification task.

- ➢ Token Preparation and Positional Encoding: Once the fusion module has done its job, the feature map received is then flattened and subjected to augmentations alongside a learnable class token with positional embedding. This ensures that the spatial context of spatial awareness is retained throughout the token series, thus enabling global attention pooling over the entire spatial context. The token sequence is normalized via LayerNorm and then passed through a feedforward projection block, which applies SwiGLU activation. This splits the feature channels for expressivity in a non-linear fashion and stabilizes representation. Dropouts are applied after every projection to prevent overfitting.
- ➢ Token Mixer: This is implemented to refine the inter-token relationships. This has batch normalization followed by a 1D convolution, which mixes information across tokens. A sense of non-linearity is introduced by GELU activation, which follows the 1D convolution. Dropouts are introduced to add regularization. All in all, this stage ensures that the token sequence is well-mixed and local dependencies are captured before attention pooling.
- ➢ Token reducer: The prime motive of this module is to learn which tokens matter. TokenReducer module is a learnable setting that computes the importance scores for each path token utilizing a minimal neural network that comprises LayerNorm and Linear layers. It selects the top-k most important tokens and elects them for further processing. This selection is done based on the importance scores it assigns itself. This adaptive approach to prioritizing some tokens over others helps focus computational resources on the most informative spatial regions, thus reducing noise and improving both efficiency and performance under limited data.

- ➢ ClassTokenAttention: This class primarily learns what to pool. It's sort of a lightweight self-attention pooling class. The selected tokens after the token reducer, along with the class token, are passed to this module. This lightweight multi-head self-attention mechanism implies class tokens as the query and the rest of the tokens as keys and values. The output of the same is a class token which is enriched with contextual information that adaptively aggregates information from the spatial hot-spots, enabling efficient global reasoning.
- ➢ Gated Residual Block: This is a simple module performing the gating task. Just as the transformer path is about to do its final classification, the class token gets processed by the GRB module. It applies a channel-wise gating vector to the class token. The learnable vector allows the model to dynamically suppress noisy and redundant channels. This gate by itself is a powerful regularizer.
- ➢ Triple-Path Ensemble Architecture

The classifier head comprises a multi-branch setup, which was introduced before; the three output heads work in parallel and are described as follows:

- Transformer Path: The visionary output from the GRB module is passed to this transformer-style MLP head, which has LayerNorm followed by a Linear layer, GELU, dropout, and another linear layer. The produced logits from this path can capture global context and long-range dependencies. This branch of the classifier head excels at integrating information from the image as a whole. It is especially effective in recognizing globally distributed patterns.
- Convolutional Path: The second path leverages the conventional paths that convolutional models use. The fused feature map from the fusion module is here processed by a 1x1 convolution classifier. This is followed by an adaptive average pooling and then flattened. The path is optimized to specifically capture high-frequency locally existing spatial features, which may get overshadowed in the global pooling system. This path is excellent at detecting the fine-grained details and local textures from the feature map.
- Auxiliary Path: The third path in the classifier head is meant to apply another section of adaptive average pooling and a linear classifier to the fused features. This branch's influence isn't dynamic like the other two branches. This path provides an indirect regularization signal that stabilizes classification. This specifically helps for samples that are ambiguous or hard to classify. The same becomes influential during the early epochs of training when the model is still learning stable representations.

- ➢ Spatial Attention and Token Selection: Just before the convolutional and auxiliary heads come into play, a SpatialAttentionGate is applied. This learns where to look in the image. This small convolutional network features a learnable spatial attention mask that highlights the most informative regions while suppressing noise. This attention masking ensures that only the most relevant spatial information contributes to the final prediction in all of these branches.
- ➢ Confidence-Weighted Fusion: The three branches get their outputs combined by the application of a dynamic, confidence-based fusion strategy. This confidence, which everything is based upon, is calculated from the softmax probability the model assigns to the actual desired class as a prediction. The same modulates the weighing of the transformer and convolutional paths. The final prediction gets computed as:

$$\text{logits}_{\text{final}}=0.4(\text{confidence})\text{logits}_{\text{transformer}} + 0.4(1- \text{confidence})\ \text{logits}_{\text{conv}} + 0.2(\text{logits}_{\text{aux}})$$

  This fusion, being adaptive, ensures that for high-confidence inputs, the model trusts its global reasoning and, in cases of ambiguity, it takes help from the local and auxiliary paths.
- ➢ Regularization and Stability Mechanisms: The model ensures its regularity via multiple regularization strategies applied. Each major transformation is followed by dropouts. DropPath has been applied to further regularize the network. Some drafts of the model tried dynamically allocating this drop probability, too, but turned out to just be an extra overhead. The GRB module introduced into the transformer path allows the model to suppress the noisy channels dynamically. Clamping is performed at various stages in the classifier itself to prevent any activation drift and maintain numerical stability even upon strong augmentation.

Each module inside SOTAQlassifierHead is built with a purpose. The CLS token enables a global context alongside positional embedding. SwiGLU and token mixer meet the need of non-linearity along the transformer head. TokenReducer and ClassTokenAttention adapt to aggregate information. Convolutional and auxiliary heads both specialize in local and regularization tasks by themselves. All these branches, followed by the SpatialAttentionGate, ensure the relevance of the features in the context of the entire image. The confidence-based dynamics actively adapt to input difficulties. This complex but comprehensive design of the classifier head enables Qonstraint to efficiently generalize and perform in low-data availability, comprising high diversity even without any pretraining or external supervision.

b) GAP layer + Linear Layer Classifier Head

This very simple and efficient head has been the globally accepted classifier head for numerous models. This neural network design derives class predictions from feature maps by applying global average pooling, which sort of transforms the spatial information by globally averaging each channel of the data and thus produces a very shrunk down informationally dense feature vector that has information from the entire input. This vector is then projected to the logits, in this case, 200 classes, via a linear layer which attaches a score to each class.

This head, though, is very simple is highly effective in its tasks. Though this has no local feature modeling or dynamic weighting, its structure ensures less computational overhead, thus making it ideal for both research and production environments. This head might sound like the bare minimum, but its utility is unmatched. It is the de facto standard in computer vision. Many well-known vision architectures have taken advantage of this classifier head. The entirety of the ResNet family, including all its major variants (18,50,152, etc.), the entirety of the EfficientNet series, MobileNet and affiliated versions, DenseNet, and many vision transformers, though primarily using CLS tokens in the architecture, use GAP and a linear layer classifier head. The list won't finish. State-of-the-art models already display their efficiency with valid computation. This classifier is an ambitious competitor for SOTAQlassifierHead to be compared against.

C. Training Setup and Implementation

1) Dataset and Splits

Any and all of the information about the model Qonstraint, which is given in this paper, comes from its application on the dataset TinyImageNet, which comprises 64x64 resolution images from 200 different classes, each having 500 images of the class. The model splits this dataset into a deterministically randomized training and validation sets (80%-20%) for each class, thus preserving class balance and fair evaluation.

2) Data Augmentation:

Qonstraint uses a two-phase augmentation pipeline. The model swaps to the stronger augmentations midway through the training phase. This helps maintain the balance of generalization with optimal convergence and early learning.

- Phase 1 (first half of the epochs): This augmentation pipeline aims to sustain clean gradients and reduce early-stage noise, something the model automatically becomes prone to upon heavier augmentation. This phase emphasizes low-variance transformations like random cropping, flipping, erasing, and jitter. This prevents over-regularizing from the start.
- Phase 2 (second half of the epochs): Once the model becomes capable of classifying the images to a respectable degree, that’s when the augmentations come with full throttle. Upon getting triggered, these challenge the model against a version of the already seen samples, which is a lot more diverse and difficult to relate to the already seen samples. This enforces invariance and discourages overfitting. All the magnitudes get stronger, and MixUp/CutMix gets increased in frequency.
- Batch-level Augmentations: CutMix and MixUp are applied using a custom collate function. Each of these is toggled probabilistically per batch depending on the current epoch phase and reward tuning. CutMix randomly picks pairs of images and combines them spatially with mixed labels, challenging the model to decide against a fraction of that image. MixUp forms linear combinations of image-label pairs to enable smoother decision boundaries.
- Validation Augmentation: These are meant to ensure reliable evaluation. All these do is center crop the 64 pixels and normalize.

3) Optimizer and Learning Rate Schedule

- Optimizer: The optimizer selected for the model is AdamW. The configuration applies $\beta_1 = 0.92$, $\beta_2 = 0.999$, alongside a weight decay of 1e-4, which helps in maintaining stability and preventing overfitting down the training line.
- Base LR Schedule: The scheduler is just a safety net for what follows. The model initializes a learning rate of 5e-4, which is meant to go down to 1e-4 via a cosine annealing learning rate, which is later manually warmed up after a set number of epochs. The code also gives an initial warmup phase in case the upcoming LR mechanisms don’t come into play. This is a safeguard against switching databases; this model is used upon. The learning rate schedule is not strictly bound to this scheduler. It is over-written by a lot of mechanisms working on dynamic control, which are described below.

4) Confidence-based control:

This novel addition takes every epoch's statistics and then performs updates based on the model's confidence. It updates the learning rate, the momentum, and the exponential moving average decay after each epoch based on validation performance, thus letting Qonstraint converge at a fast rate after early instability gets settled after a few epochs. The operations are listed as follows:

a) Reward and confidence: Every epoch is followed by a reward parameter, which is computed as the difference between the validation accuracy pre- and post-epoch and clamped to a given range to maintain stability.

$$\text{Reward} = \text{val\_top1} - \text{prev\_val\_top1}$$

The model updates its confidence by primitively applying a sigmoid function over the reward derived.

$$\text{Confidence} = (1 + \exp(-0.5*\text{reward}))^{-1}$$

This sigmoid function assigns a value of maximum magnitude 1 to the confidence value, which is then slightly modified within the training loop, serving as a guide for the subsequent adjustments.

b) Learning rate control: The learning rate is governed by a dynamic and overlapping schedule that acts in layers.

- LR boosting: if the reward is positive and the confidence exceeds 0.7, the model's learning rate is increased by a factor of the reward to help it escape lower minima. When this occurs early in training, it can accelerate convergence. The same approach is kept active during the warmup phase, which may cause the warmup to end prematurely, but this is not harmful and simply aids convergence.

$$\text{LRnew} = \min(\text{MAX\_LR}, \text{LRold} \times (1 + 0.065*\text{reward}))$$

- Plateau reset: if no LR boosts occur in the last "patience" epochs (set 4), and the validation accuracy hasn't seen any growth in the last 4 epochs, the model tries to force itself out in a last-ditch effort out of the minima or saddle point by boosting LR regardless of what the model confidence is telling it to be.

$$\text{LRnew}=\max(\text{LRold},0.6\times\text{MAX\_LR})$$

- Manual Cosine restarts: Beyond the scope of the RL-based interventions. The code also implements a manual cosign annealing schedule with warm restarts. This is applied whilst ensuring that the boosting or plateau resets are not triggered by this adjustment, and the learning rate is still able to decay after it. This is set in a way that boosts the model out of suboptimal minima observed whilst training.
- Fallback: If the learning rate neither gets boosted nor is reset, a manually specified cosine decay acts on it, which also stimulates restarts. This is the conventional cosine warm restart itself, just specified manually for hyperparameter tuning. As stated, this is subordinate to any other LR adjustments. It gets overshadowed by any other LR adjustments and just acts as the base for the LR, ensuring the model decays its LR slowly overall.

c) Momentum Tuning: When the training is experiencing a phase of improvement, the momentum is increased by a factor of 0.02*reward but capped at 0.98 or 0.995, depending on what stage the training is in. When the learning encounters a local or global plateau, the momentum is decreased by 0.01 per epoch but floored at 0.85 at least. This dynamic adjustment helps keep the convergence fast and reliable, ensuring learning effectiveness and stabilizing the training when progress stalls.

d) EMA Decay Oscillation: The exponential moving average decay rate is dynamic and is adjusted using the following equation:

$$\text{EMA}_{\text{decay}}=\min(0.999, \max(0.90,0.96+0.02*\text{confidence}))$$

This setup ensures that the EMA is tracking the model rather closely during the phases of fast learning when the confidence is higher, and eases out when training is stable, observing a lower confidence level. This provides both regularization and a sense of robustness to the weights.

e) Gradient clipping: The model applies gradient clipping with norm capped at 1.0 during its back propagation phase, which prevents exploding gradients, something the training is prone to during initial unstable epochs or after sudden LR boosts. This is necessary to maintain stability during rapid convergence in deep models utilizing dynamic schedules.

f) Stochastic Weight Averaging and hardness prioritization: During training, every sample's loss is given weightage by its hardness, determined by the model's confidence in its prediction. This helps the model focus its learning on harder samples. SWA is applied conventionally without any tweaks to the setup and is triggered after 85% of training is done.

5) Model Weight Reloading:

The LR and momentum being so dynamic automatically come with the issue of instability in training and unpredictable absolute accuracy values. This is a strategy included in the training loop to safeguard against performance drops that occur after aggressive learning rate interventions or periods of instability. The saved checkpoints are reloaded at pre-defined epochs with the best weights so far, and that happens via a trigger multiple times through the training, especially in areas of the model being very prone to instability. Reloading prevents sudden divergence at critical stages and helps recovery over stability. Any catastrophic forgetting is taken care of, ensuring that the model does not lose previously acquired knowledge. All this helps in a higher final validation accuracy, forcefully applying heavy convergence towards the end, which helps recover from explorations towards the end.

6) Regularization and Loss:

- Label Smoothing: The label smoothing factor is not static. The model uses a higher label smoothing factor when the model is early in the training. This is the region where the model is unstable. As the model progresses, it reaches its floor value of 0.05 at 75% the training epochs. [max(0.05, 0.2 * (1.0 - confidence))]. This dynamic regularization prevents overfitting early on and sharpens the predictions as the epochs progress.
- Auxiliary Loss: This auxiliary classifier's loss weight starts at a higher value of 0.4, which later decays linearly down to 0.2 as the training progresses. This is done in the training path itself and ensures the supervision of the auxiliary head is higher at the start, as explained in the classifier head, and gradually outputs its main classification objective after the model is in a stable state.

D. Batch Size and Hardware

The training is conducted on the publicly available GPU P100 by NVIDIA over Kaggle. The same is selected for it balances compute power and memory capacity with versatile applications and a reliable benchmark. The model uses a single GPU of such kind and a batch size of 256, which balances VRAM utilization and training time efficiency. Mixed precision (AMP) is also used throughout the training. This leverages half-precision float pointing operations in every step possible, which increases training speed and memory consumption whilst maintaining numerical stability and accuracy.

The training loop takes 5:33 per epoch, and the entire model is trained in under 7 hours, which is a remarkably short duration for what it achieves and from where it achieves it, thus illustrating the efficiency of the implementation of the model. The same uses just 9.2 GBs of VRAM. Even though the model caters to 96.5M parameters, the architecture is set in a way that makes it really fast in convergence and is still efficient in VRAM. A checkpoint of this model is thus weighted at 2 GBs, but is fast in operations involved.

E. Reproducibility

To ensure deterministic results and reproducibility of the model, all random seeds are fixed, including Python's built-in random module, NumPy, PyTorch, and CUDA backend. The data loaders are seeded too, and that too individually ensuring data shuffling and augmentations are consistent. This helps to avoid subtle variations in training due to variable data loading parameters.

The model also saves checkpoints whenever it reaches a new personal best in top 1 validation accuracy, which preserves all model weights, optimizer state, AMP scaler, EMA/SWA states, current epoch, and best validation accuracy, thus ensuring the training can be resumed with everything in the same state. The saved checkpoint will be of the best validation accuracy until that point, and that will be the one automatically selected for evaluation and deployment, thus ensuring that the reported results are in the checkpoint saved.

F. No Pretraining or Resizing

Qonstraint was always developed with the motive to not use any external data or pretrained weights and layers, to portray how effective the model is by itself. Alongside not using any pretrained data, and all weights being initialized from scratch, the images are neither downgraded nor upgraded from their native 64x64 resolution. This isolates the inductive bias of the architecture and the effectiveness of the training pipeline, thus ensuring that all the performance gains are solely because of the model and innovations involved in optimizing it, rather than any input modulation or supervision.

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 13 Issue VII July 2025- Available at www.ijraset.com

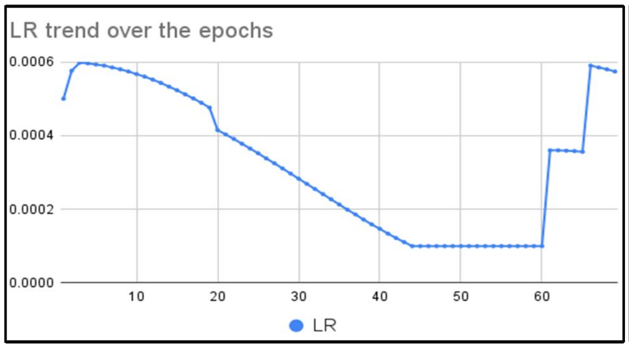
IV. EXPERIMENTS

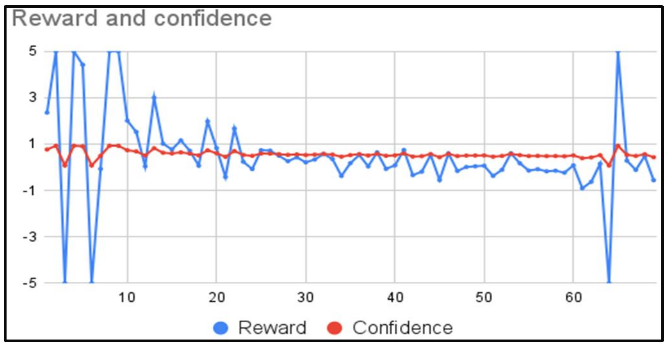
A. Qonstraint 5.9: Metrics and Comparison

The methods described above look great on paper, but they need some numbers backing them. This section will provide numbers and achievements to the noble and efficient approach of Qonstraint 5.9. The model was running on TinyImageNet, as told before, with a batch size of 256 on training and 512 on validation. The following reports the data on the second classifier head, which will be analyzed in the detailed ablation study that follows this section. The parameters for the optimizer with AdamW were 0.92 and 0.999 for the two betas with a weight decay of 5e-5. Learning rate started at 5e-4 and was run down to 1e-4 following the confidence-based approach described. The 70 epochs utilized AMP and dynamic label smoothing by a factor of label_smooth. SWA was enabled 85% into the training, and the EMA decay rate was variable from 0.96 up to 0.999, again based on confidence.

1) Confidence trends and dynamic control

The LR as described follows both, a cosine annealing manual warm restart and a RL based adjustment which gives the LR such a dynamic shape, which self optimizes convergence. The logs show that the model crosses 40% validation top-1 at just epoch 40 without any harm to other metrics, thus adding to the benefit of this dynamic LR cycle. The late push in the LR lets the model explore for the last 10 epochs before terminating.

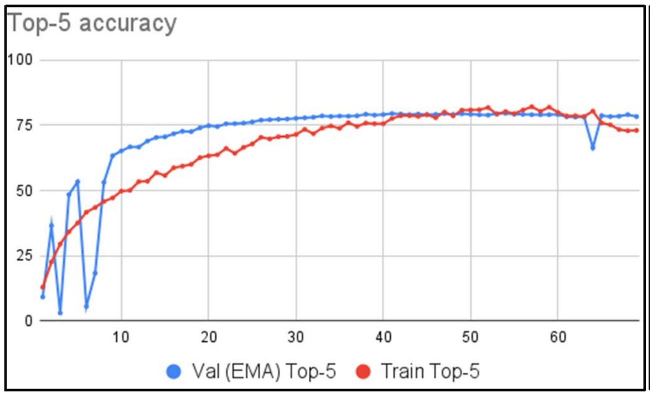


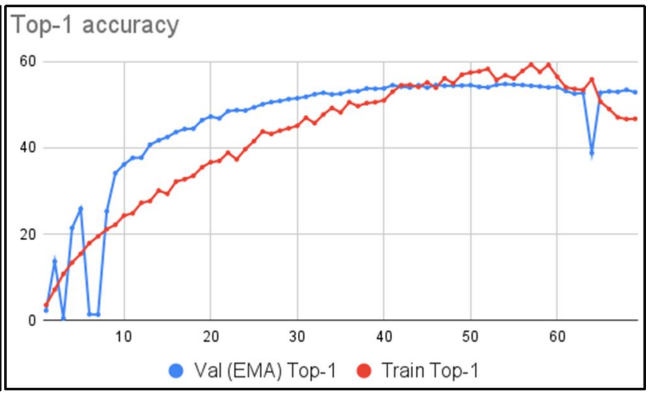


The reward signal responds to the difference in validation top-1 between the epochs, and confidence responds to the reward. The plot shows the response of confidence to the clamped reward. The confidence acts as a buffered reward whose calculation is inherently based on the reward itself. Momentum and EMA decay are also kept dynamic, and the logs will portray the same. Even though the numerical impact is minimal, it adds up over the epochs and is thus evident too.

2) Accuracy trends, validation, and training

The model was run 3 times, and the results are thus certainly reproducible and show a similar trend overall. The accuracy trends are very promising and show excellent results. The plots show the convergence over the epochs, the same is rapid in early training, facilitated by the dynamic controls. Top-5 accuracy peaks faster than top-1 accuracy, as expected, showing that the model was uncertain but still somewhat correct with its predictions in early training. Training accuracies show a monotonous gain till epoch 60, after which the model was forced to explore, the same peaked between 59 and 60% consistently for top-1 and 81-82% for top-5. These peaks were after the model hit the peak validation accuracy, they were at a modest 57 and 80 percent at the peak validation epoch. The low variance in this data adds weight to the model's results not being a one-time luck.



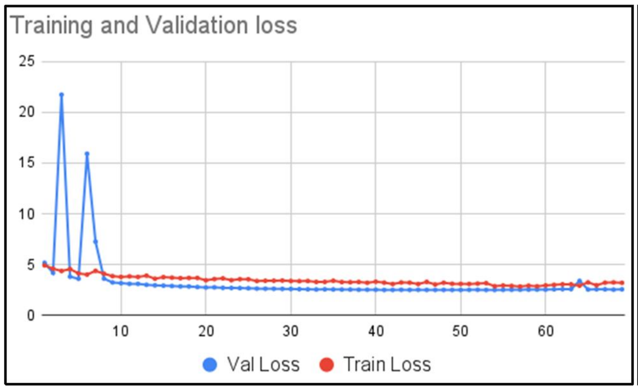


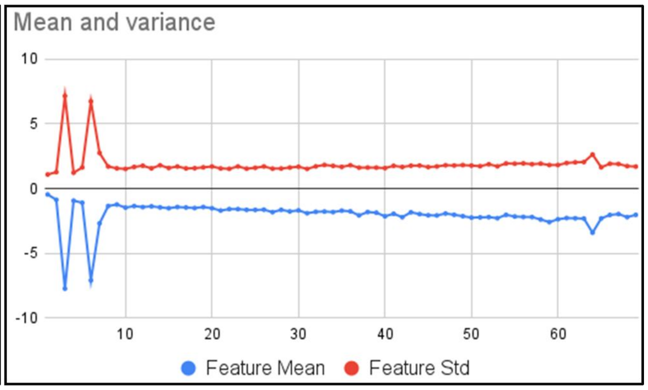
The peak validation epoch showed promising results. The model consistently crossed 54.5% and almost reached 55% accuracy each time. Top-5 accuracy was around 79.5%. The models hit these accuracies before the 60th epoch each time, thus showing promising convergence, and the LR reset was optimally placed so as to not hinder learning but avoid plateaus, needing the model to be time efficient as well.

Metric	Value (Mean ± Std)
Peak Train Top-1 (%)	59.5 ± 0.4
Peak Train Top-5 (%)	81.2 ± 0.5
Peak Val Top-1 (%)	54.7 ± 0.2
Peak Val Top-5 (%)	79.5 ± 0.3
Epoch of Peak Validation	54 ± 2

3) Loss, Mean, and Variance

We notice that the validation loss, after facing early instability, remains beneath the training accuracy, which shows the effects of the augmentation on the training accuracy. Even though the model gained accuracy in training, it was facing higher loss values, which shows that the regularization and augmentations were doing their thing. The mean and variance also remain in a narrow bracket, which too almost mirrored by each other, thus adding to the model's consistent behavior. Whenever the mean goes out of the narrow band, so does the variance, thus indicating that the model knows that the optimal weights are in a well-defined band from this mean, thus recovering from the loss.





4) Summary of trends over epochs, late training trends

Epoch	LR	Confidence	Momentum	EMA	Train top-1	Train top-5	Val top-1	Val top-5
1	0.0005	0.7645	0.9471	0.9573	3.62	12.97	2.35	9.22
10	0.000567	0.7315	0.98	0.9746	24.34	49.76	36.16	65.14
20	0.000415	0.6017	0.98	0.972	36.67	63.27	47.23	74.78
26	0.000338	0.5884	0.98	0.9718	43.77	70.33	50.08	76.92
41	0.000134	0.5915	0.98	0.9718	52.99	77.54	54.48	79.44
54	0.0001	0.5212	0.9834	0.9704	56.79	80.17	54.76	79.55

The epoch summary listed above is a de-magnified image of the convergence up to the peak accuracy range after which the model plateaus in that accuracy range itself for a few epochs after which the LR restart force exploration, even after hitting high LR values, the model still finds a stable point ready to learn more for any gains if possible. The same is met with subtle instability, which is later catered to. This phase of the learning hasn't been optimized and leaves room open for future improvements. The same has still been tested upon and gives fairly consistent results as follows.

Epoch	LR	Confidence	Momentum	EMA	Train top-1	Train top-5	Val top-1	Val top-5
60	0.0001	0.51	0.9691	0.9702	56.54	79.89	54.05	79.06
65	0.000356	0.9241	0.995	0.9785	50.65	75.83	52.77	78.64
68	0.00058	0.5597	0.995	0.9712	46.64	72.85	53.42	79.02
69	0.000574	0.4305	0.9838	0.9686	46.71	73	52.86	78.27

5) Comparison with other models

Model	Val Top-1 %	Params (M)	VRAM	Training Time	Hardware Used	Speed vs. P100	Ref.
Qonstraint 5.9 (Ours)	54.8	96.5	9.2 GB	< 7 hours	Tesla P100	1.0× (base-line)	This work
ResNet-20	55.4	0.27	~8 GB¹	~14 hours (10 epochs)	Tesla V100¹	1.5×	[10], [6]
ResNet-18	52.7	11.7	~9 GB²	~7 hours (10 epochs)	Tesla V100²	1.5×	[16], [6]
ResNet-34	57.5	21.8	~10 GB²	~9 hours (estimated)	Tesla V100²	1.5×	[16], [6]
DenseNet-121	56.1	7.0	~4 GB³	~4 min/epoch (~6 hrs est.)	GTX 1080 Ti³	~0.8×	[9], [1]
DenseNet-BC (k=12)	60.0	0.80	~4 GB³	Not reported	GTX 1080 Ti³	~0.8×	[9]
VGG-11	53.0	132.9	~10 GB⁵	~1.5–2 hrs/epoch	RTX 2080 Ti⁵	~1.8×	[17], [9]
VGG-16	54.0	138	~10–12 GB⁵	~1.5–2 hrs/epoch	RTX 2080 Ti⁵	~1.8×	[17], [9]
SqueezeNet	20.5	1.25	~3 GB	~6–8 hours	Unknown (mid-tier)	~1.0×	[18]
GoogLeNet (Inception v1)	48.1	6.6	~4–6 GB	~8 hours (estimated)	Tesla K80⁷	~0.5×	[19]
AlexNet	41.8	61.0	~4–5 GB	~6 hours	Tesla K80⁷	~0.5×	[17]
ShuffleNetV2 0.5×	49.6	1.4	~3 GB	~5 hours	Unknown (edge GPU)	~1.0×	[20]
MobileNetV2	51.3	2.3	~3–4 GB	~5 hours	Unknown (mobile)	~1.0×	[21]
PyramidNet-110	58.3	1.7	~6 GB⁶	~10 hours (10 epochs)	Tesla V100⁶	1.5×	[22], [6]
WideResNet-28-10	61.7	36.5	~12 GB⁶	~12 hours (10 epochs)	Tesla V100⁶	1.5×	[23], [6]

Note: Tesla V100 is about 1.5× faster, A100 is about 3×, and RTX 3090 is about 2× faster than P100. Wherever the exact GPU is unspecified or TinyImageNet numbers are unspecified in references, best-effort estimates or GitHub applications are used.

B. Ablation Study: Lighter Convolutional Stem

This ablation study aims to examine the efficiency of the Qonvstem setup that the model uses. This is the baseline over which QonvViT blocks then work, which will be analyzed in another ablation study. This section performs a raw image transformation on the data, making it suitable for deeper models. This component in Qonstraint applies multiple convolutional layers, each of which has varying strides as analyzed as they down-sample by their stride. Both of the Qonvstem were trained in a very similar environment, both using the SOTAQlassifierHead as discussed, which leverages recently found techniques, thus portraying compatibility with upcoming research work. keeping everything else constant. Their epoch-wise metrics are summarized below to highlight the practical effects of stem complexity, which are very visible in the early training itself.

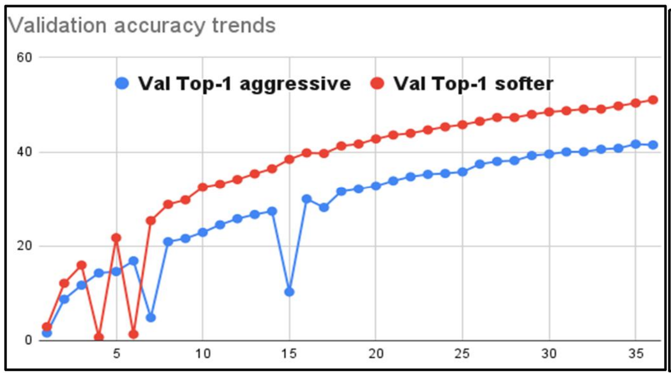
1) Overview

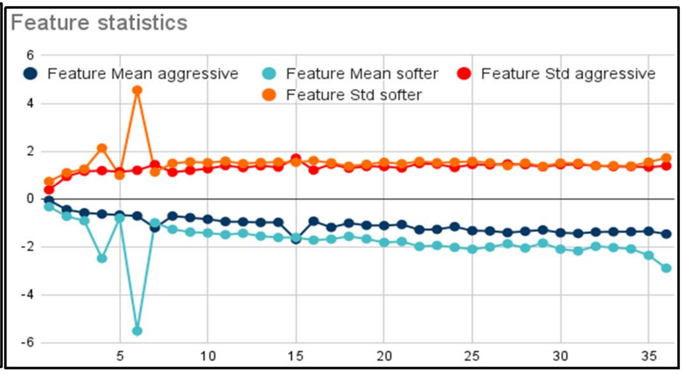
a) Aggressive Qonvstem: This is the Qonvstem, which outputs a lighter feature map, that is, it down-samples more. The first two convolutional layers both use a stride of 2, thus reducing the size from 64x64 to 16x16 within the first three layers. This outputs a very compact low resolution feature map. This is very easy to compute and thus can cater to larger batch sizes and way faster training at the expense of fine-grained information. This efficiency in numbers is observed, with training time per epoch taking less than 2 minutes and 30 seconds, while operating on a large batch size of 512 and still consuming only 6.5 GB of GPU, at a utilization rate of less than 80 percent of the computational capacity.

b) Softer Qonvstem: This Qonvstem is the one the model possesses; this delays the spatial downsampling and does it after the first two convolutional layers have done their job instead. The third layer then reduces that 64x64 to 32x32 after having the first 2 convolutional layers bring in a sense of neighborhood identity. This higher feature map inherently means that it is computationally more demanding but retains more detailed information through the downstream processing, which thus improves representational capacity and thus the accuracy of the model, whilst still being computationally feasible. The same takes around 6 minutes 10 seconds per epoch for the training time, on a batch size of 256, and takes around 9 GBs of VRAM.

2) Training and Validation Dynamics

In early epochs, the aggressive Qonvstem showed slower but steady improvement whilst validation accuracy also dropped down amid things, but was still followed by quick recovery in learning boosts. The softer stem showed the benefit of richer early feature extraction thus giving higher accuracies. The drops were even sharper showing instability in early epochs but its expected when working with deeper dynamics. In the mid-to-late training the aggressive Qonvstem showed consistent increase in both training and validation accuracy thus showing an optimal setup in itself. Feature statistics also remained stable indicating optimal feature learning even at reduced resolutions. The softer stem gains higher accuracies but with higher variability between epochs, the same is understandable as it is a subject to higher variances thus more dynamics to observe.





3) Performance Trend Analysis

The aggressive Qonvstem led to a very compute-efficient training regime, which is very well suited for environments that need rapid prototyping and heavy resource constraints. The softer Qonvstem also, though being heavier in compute than the softer Qonvstem, showed a higher accuracy standard overall, thus implying weight to its relatively higher compute.

The two Qonvstem show two approaches, one which offers a balanced approach, delivering strong performance even by cutting large fractions in training speed and compute needed, whilst the other provides higher absolute accuracy by carefully managing training stability and resources.

C. Ablation Study: Effect of QonvViT Depth

This ablation study is meant to understand the subtle impact of model depth and width on the behavior of the model. This is done on a draft of the model, which is fairly similar to Qonstraint 5.9 but wasn't optimized point by point to the dataset, thus nullifying any effect of those optimizations and the gain being purely just by depth.

The depth in this context refers to the number of QonvViT blocks used in the backbone per stage. The embedded dimensions are proportionate to the width of the blocks involved at each stage, directly affecting the model's capacity.

All the models were trained in the same conditions, with the only parameter being varied being the depth of the QonvViT blocks. All of them used SOTAQlassifierHead and ensure a direct comparison of the model.

1) Overview

- Lighter: embed_dims= [64, 128, 256, 512], depths= [3, 4, 5]
- Heavier: embed_dims= [64, 128, 256, 512], depths= [3, 5, 6]
- Heaviest: embed_dims= [128, 256, 512, 1024], depths= [4, 4, 6]

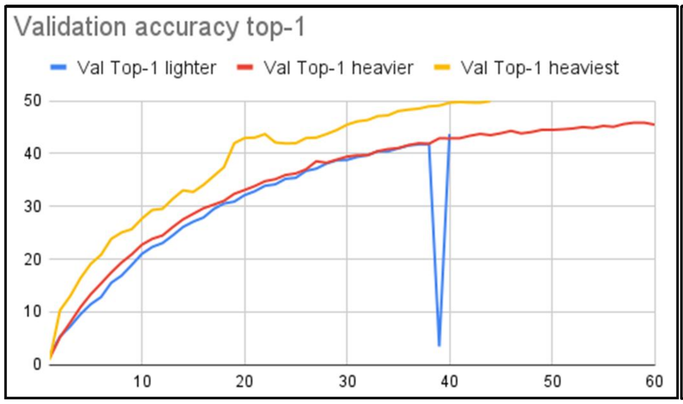
The above explains what we'll be testing in this section. These have a descending order of computation required; the numbers of the same will be described in what follows. The three having the same architecture just vary in depths and widths, as the two lighter ones have varying depths, and the heaviest one has increased embedded dimensions too. The depths not being kept the same will show the effect of non-linear depths when plotted, though the effect of the depths will be the same, as the heaviest also uses 14 blocks.

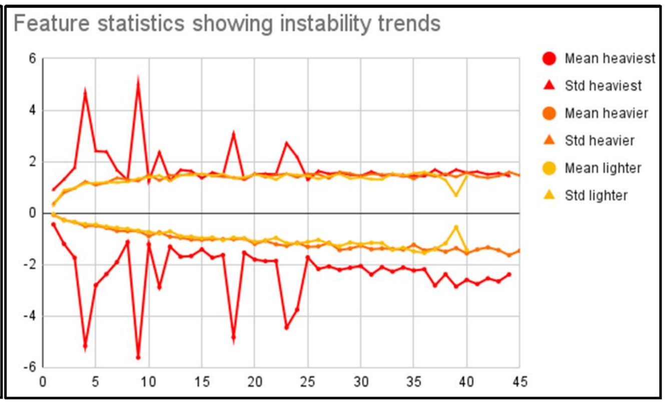
2) Training and Validation Dynamics

a) Lighter Model (uses 10.6 GBs at 512 batch size), 3 minutes 35 seconds per epoch: This improves at a good pace, has lower initial and peak accuracies than the other two models. Regardless of the accuracy, though, this showed the smoothest training and validation curves, with minimal oscillations and drops in the data collected. The feature mean and deviation remain within a very small range, indicating consistent feature extraction without instability. There is just one exception in which the LR cycle caused instability.

b) Heavier Model (uses 11.2 GBs at 512 batch size), 3 minutes 50 seconds per epoch:: Shows good early and mid-epoch improvement. The validation accuracy is consistently higher than the lighter model but slightly below the heaviest one. Loss and accuracy for this are relatively stable with moderately intense oscillations. The feature statistics also show a lot more stability compared to the heaviest model, thus indicating a good balance of capacity and regularization.

c) Heaviest Model: (uses 9.9 GBs at 192 batch size), 8 minutes 10 seconds per epoch: This achieves the fastest gains early on as it has the most headspace to inculcate new information too. The increased headspace also enables this to achieve the highest validation top-1 and top-5 accuracies across all models. This increased headspace results in more pronounced fluctuations in accuracy and losses, especially in later epochs, making it sensitive to overfitting and optimization instability. This also resonates with the widest range in mean and standard deviation, which reflects the model's high capacity.





3) Performance Trend Analysis

The heaviest model excels in representational power and thus accuracy, but at a large cost of resource demand and training instability. This is suited for scenarios where maximum performance is required. The other two models cater to resource constraints, and both are sort of similar in performance too, given that heavier models converge faster, having more depths. Qonstraint 5.9 aimed to leverage this benefit of the heaviest model being representation-enriched and then tone it down towards the size of the heavier and lighter models. The same lets Qonstraint be so efficient as seen in the later drafts of the model, but this ablation study was a big contributor in deciding and thus now finalizing the model setup, the same uses. The optimal configuration, thus finally opted in any scenario, shall reflect the thought given to deployment context and efficiently balance accuracy requirements against resource constraints.

D. Ablation Study: Effect of width and model optimizations after structure definition. Qonstraint 5.5,5.6,5.9

This study is meant to show how the model has grown up to what it is right now. The same has been portrayed in three stages of the model, which depict substantial growth and are ablation-worthy. The first model is Qonstraint 5.5; this will be the baseline of Qonstraint. The second one is Qonstraint 5.6, which is similar to all techniques used in 5.6, with the only practical difference being an optimal balance of depth and width of the model, thus giving improved results. And the third is an almost replica of Qonstraint 5.9, just a bit older variant. This is one of the models reported. Point to be noted that 5.5 to 5.6 wasn't a 1 draft jump, there were many drafts in between, the best of which was selected as 5.6. The same applies to 5.9. This has the same width and depth as 5.6, but many practical optimizations have been made in the model itself. This ablation study will show how increasing depth and tweaking techniques can make the model perform better.

1) Overview

a) Qonstraint 5.5: Baseline Model, Minimalist Capacity: 5.6 will serve as the foundation of this ablation. This architecture is intentionally smaller, being one in which the model was still being decided. This later on turned out to be the best one, and then I used it forward. This uses embedded dimensions in multiples of 2, starting from 64 and 8 blocks in increasing depths per stage through the three stages. All normalizations in the backbone and the SOTAQlassifierHead use BatchNorm, which, whilst being good for training stability, often compromises on expressiveness in transformer-heavy networks. Data augmentation is rather conservative. It implies standard image augmentations and MixUp, but doesn't include the aggressive techniques of CutMix and RandAug yet. A Cosine annealing LR schedule and both EMA and SWA are observed.

b) Qonstraint 5.6: Capacity increased from Qonstraint 5.5: This is architecturally identical to 5.6 in every aspect to 5.5, just the crucial difference being the optimized and tested embedding size and depth. The channel widths for this are doubled at every stage, reaching 5.5, which then matches the configuration we have used for Qonstraint 5.9. Thus, this makes up to be a very profound middle ground between the two other models, bridging the gaps in two separate leaps, thus making up for an ablation study. This change alone dramatically increases the model's representation power, as seen in the last ablation study. This change has been put alongside in this ablation study also to show the effects of the changes made up to 5.9, about this big change. All aspects other than embedded dims were kept intact, and what follows is Qonstraint 5.6.

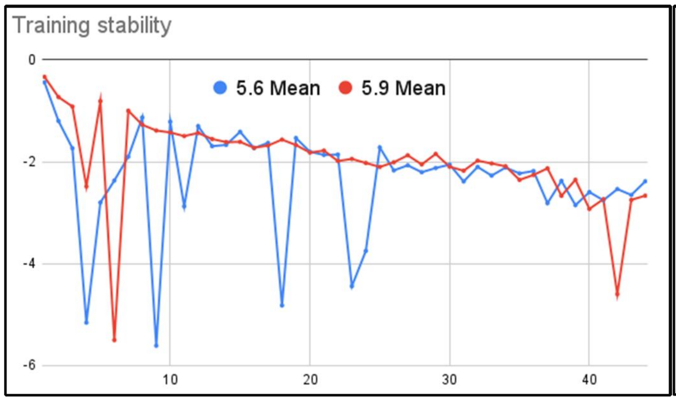
c) Qonstraint 5.9: The 9th version of the 5th concept of Qonstraint demonstrates a deliberate push to maximize generalization and representation over the added benefit of more embedded dimensions, as seen in Qonstraint 5.6. Whilst keeping the dimensions and depths intact. The new observed differences are due to the architectural changes made rather than just pushing with compute, the same which makes Qonstraint so efficient. The backbone and classifier now use LayerNorm throughout, biasing performance towards transformer-like architectures utilizing heavy augmentations. SOTAQlassifierHead was already there in previous versions, but the numbers weren't optimized. Now the augmentations also incorporate CutMix and RandAug, which expose the model to a way more challenging distribution of inputs during training. Now, the DropPath probabilities are also dynamic alongside the other dynamic parameters. EMA and SWA are further optimized to give better performance on the given structure.

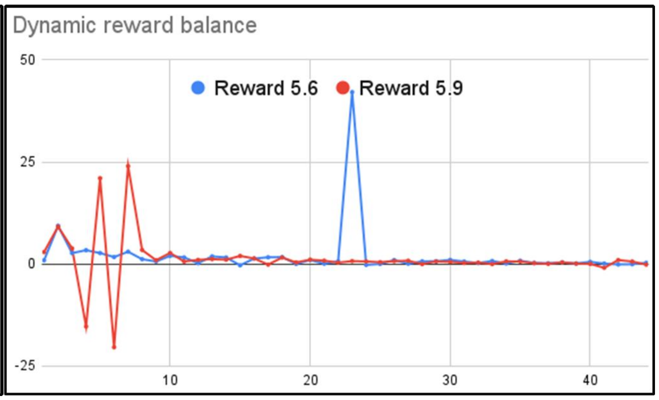
2) Training and Validation Dynamics

The dynamics of the three models show distinct patterns in learning and generalization throughout the training period.

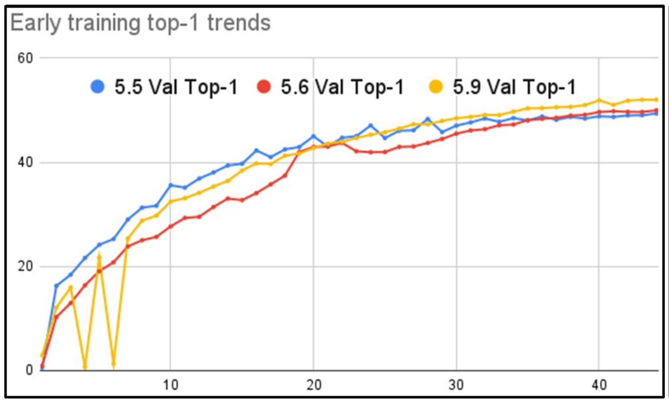
a) Qonstraint 5.5: The training of this model has very stable learning curves, given the application of BatchNorm. The same is accompanied by very low variance in feature statistics and steady improvement in accuracy. Though not having the benefits which the other two models possess, this has a lower ceiling of accuracy given narrow chancel widths and sub-optimal parameters and regularization. Still gives a strong benchmark to compare other models against. This progresses slowly but reaches a plateau around 48.9% top-1 and 73.9 % in the top 5 peak of validation accuracies. This curve though, is really smooth without any abrupt spikes or drops observed. Thus portraying the model's conservative architecture which shows the model's priority towards stability over exploration in this stage.

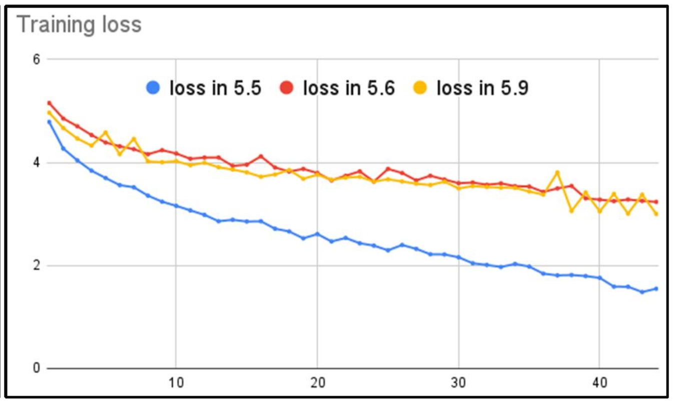
b) Qonstraint 5.6: The improvement in channel widths shows a clear and significant improvement in accuracy over the narrower model. The convergence and final accuracies are both beaten. The learning curve still remains smooth, but the feature statistics are varied as shown in the graphs, the reason being the lack of optimizations in the training line. Even though it has not been optimized with optimal augmentation, normalization, and classifier variations, it still poses up to give good performance statistics and a good improvement overall. The validation accuracy increases with higher convergence, crossing the 50% benchmark at epoch 44 onwards. The increased embed sizes enable faster and more effective learning. The top-5 accuracy also surpasses 76% accuracy, thus showing a notable improvement over the baseline. The curve also remains smooth and stable with a minor variance increase against 5.6. The higher capacity shows good results in itself. The convergence on this, though, is relatively lesser, as the same configuration of 5.5 is subjected to an exponentially larger headspace to learn in, thus taking more time. The accuracy still shows minimal improvements after this, but it isn't the subject of this ablation study, as early training shows the applications of the claims made.





c) Qonstraint 5.9: The stability gained in mid-training with the changes made is evident. The changes made collectively result in a model which not only learns faster, achieves higher accuracies, and gives a better performance overall, but also shows more pronounced fluctuations in training and validation, thus showing the practical double-edged effects of this high capacity and heavy regularization and augmentations whilst keeping everything dynamic. Feature statistics show the widest range, especially at the start. The model later converges. The careful tuning ensures the model doesn't end up overfitting a lot and remains stable for long, and the payoff is clear. The models having similar compute sizes also let us see the rewards controlling the means per epoch. The graph shows how the dynamic rewards don't overreact to the means fluctuating around. This achieves the fastest and highest gains of anything, reaching over 52% accuracy in a similar number of epochs. The initial epochs show a way sharper rise, thus reflecting the benefits of the enhancements made. Top 5 accuracy also follows peaking up to 78.1% which is the highest among all models, and sustains its lead throughout the training. The learning curves show fluctuations throughout the training, which come as a byproduct of high capacity and aggressive augmentations and dynamics. The overall trend is positive and dominant.





3) *Performance Trend Analysis*

The shift of embedded dimensions showed that increasing capacity by itself is a clear gain with a modest reduction in validation loss. The architectural and training innovations that then followed reveal that innovations in normalization and augmentation provide a larger additive boost. Qonstraint 5.5, though not performing as well, brings about the insights of training stability, where stable training is paramount. The same can be optimized in that direction if the scenario demands predictability. The hierarchical upgrades noted in this manner showcase the importance of both and the balance a model shall inherit.

E. SOTA Qlassifier Head vs GAP + Linear

This section is meant to discuss the biggest debate the model poses. The model possesses a novel classifier head, which has been discussed before. It is set in a way that is easy to substitute with the conventionally optimal GAP + linear classifiers, which have proven to be good at their task and thus have been widely opted for too. Models like ResNet, EfficientNet, vision transformers, MobileNet, DenseNet, all use a setup in which a GAP layer is followed by a linear layer, thus showcasing their practical supremacy in this domain. This analysis is done to understand how these two architectures, with vast dissimilarities, are still able to produce similar results, thus showing the practicality of the SOTAQlassifierHead.

The GAP head leverages global average pooling followed by a linear transformation, which gives scores to each class. This approach, being fairly simple and efficient, makes it a default choice in almost all the renowned architectures. The SOTAQlassifierHead, on the other hand, is a lot more sophisticated; its architecture was explained before. It uses an ensemble of pathways and confidence-based dynamics that aim to enhance the head's generalization and representational capabilities. These two approaches are thus fairly different and worth a detailed study.

The study that follows will be testing both of these classification heads over Qonstraint 5.9. The study is performed whilst tweaking two very essential parameters:

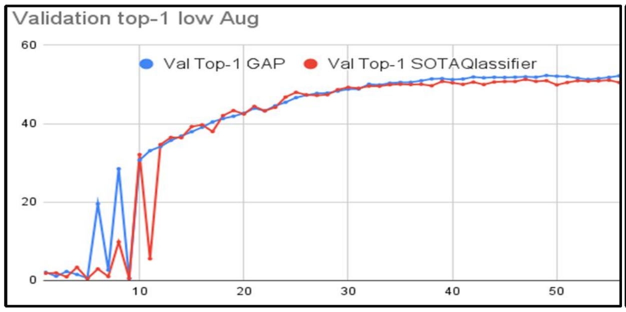
- Augmentation: The performance of the two heads is assessed over three different levels of augmentation, each of which will be described when they are encountered in the study. This will help us assess if the two heads are individually sensitive to the augmentation, too, or if the augmentation makes no big difference and the model architecture does its job.
- Depth: The two models will perform in two depth environments, one shallower and one a bit deeper. The depth ablation is being talked of again and again because it's a really big component to optimize when setting up a model this deep and sophisticated. This particular study shows how the two heads interact with that depth, catering to model complexity and overfitting issues.

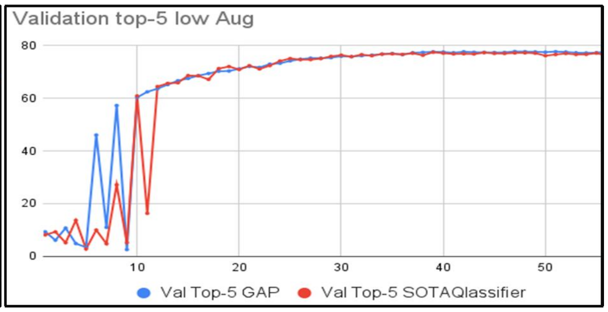
This study alone comprises data of over 100 hours of runtime over Qonstraint, which takes around 6 hours of runtime, and thus is very detailed in its results and observations.

1) *LOW Augmentation*

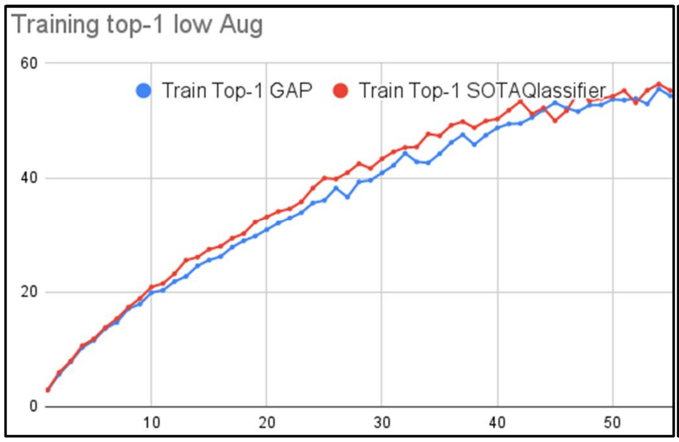
This augmentation pipeline will test both models against minimalistic levels of augmentation. Both the heads had been trained for 60 epochs, after which both showed plateauing results. This level of augmentation was tested only on the higher depths, numerically 2,3,5 per layer. Depth analysis will be seen in the other augmentation pipelines, which gave better results. This study will be based only on the differences between the classifier head and the raw brute force of the heads themselves.

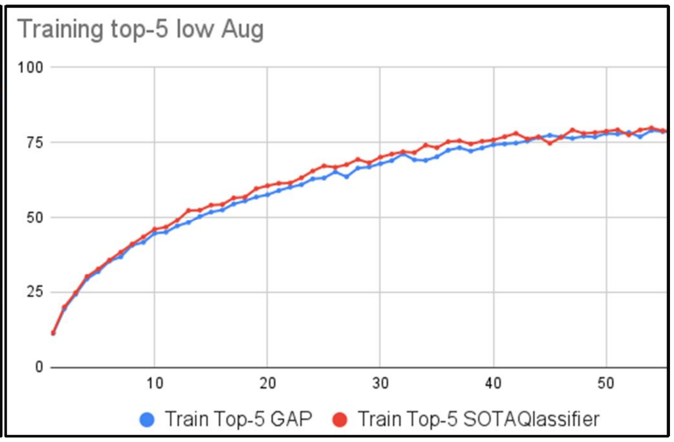
The augmentation pipeline still has 2 phases. The first phase crops down to scale 0.95, whilst the second goes to 0.9. Horizontal flips are still seen in both, but RandAugment is exclusive to phase 2. Color jitter values go up from 0.1 to 0.2 as the phases transcend, and the RandomErasing probability is doubled too, from 0.05 to 0.1, following a scale of (0.01,0.1) to (0.02,0.15) for the two phases, respectively.





Both models perform fairly similarly in this augmentation regime, both experiencing a case of subtle overfitting. Validation statistics for GAP peak out at 52.28% top-1 at epoch 49, and whilst Qlassifier peaks at 51.29 at epoch 47. Top-5 for both plateaued at just above 77%Training statistics for both are fairly similar, both peaking at epoch 54, Qlassifier being a sliver above 56, whilst GAP being a sliver below.





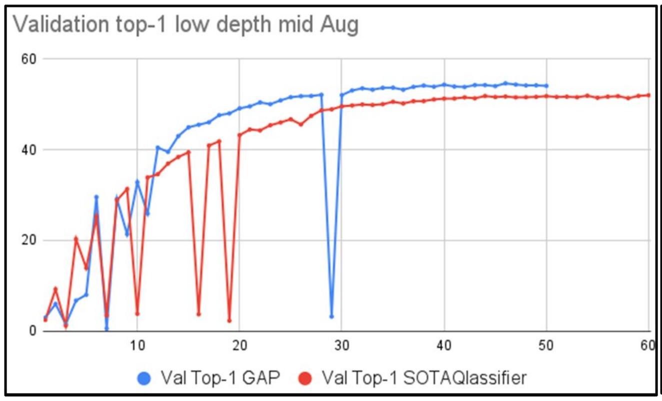
Low augmentation made both the heads tower up to similar values in a similar fashion, SOTAQlassifierHead having a convolutional bias at low confidence, learnt the data just a bit faster at volatile regions, making it hold up to GAP, which is meant to be a lot more efficient. Minimalistic augmentations had to induce overfitting, and both the models behaved similarly to a forced overfitting regime, which doesn’t promote understanding, just makes the model mug up the data itself. The backbone of Qonstraint did its job of regularizing, and we see that both models are not going towards harmful biases towards the training data.

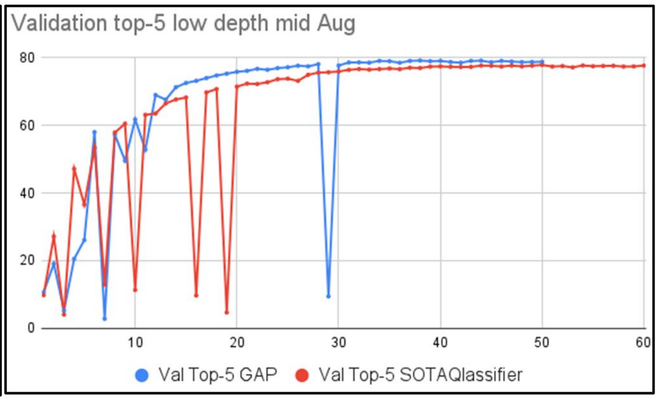
2) MID Augmentation

Now we talk in the context of medium-level augmentations. This pipeline is a bit more ablated than the low augmentation one. The random cropping is now twice as harsh for both phases, going from 0.9 in phase 1 itself to 0.2 in phase 2. The horizontal flip works the same way still, but the jittering of colors now happens to a degree of 0.2 and 0.3. Random erasing of pixels is heavier in phase 2 with a probability of 0.15. Random augment application still stays the same. This is sort of a middle ground between the two other augmentation regimes, giving potentially the best results out of all as a result. The following study will explain the same.

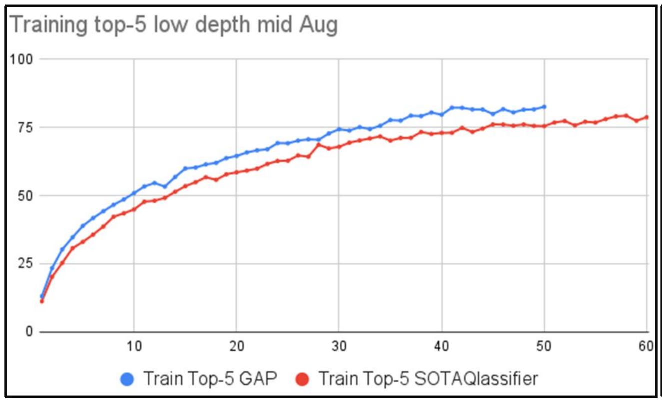
a) Low depth

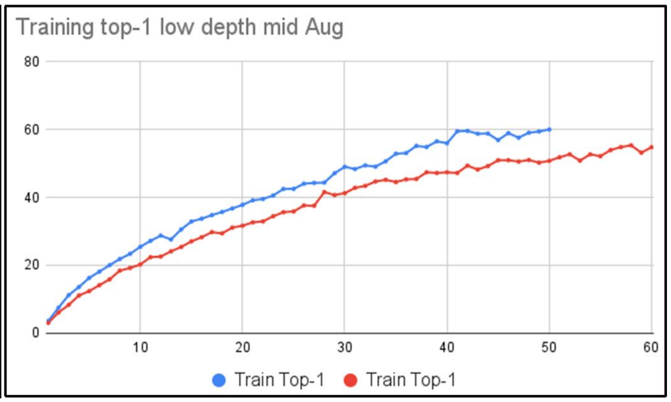
Looking at the validation accuracies, we see that GAP peaks at 54.66% accuracy in the 46th epoch, whilst the SOTAQlassifierHead caps out at just above 52% validation top-1 accuracy reached at epoch 60. The graphs beneath show the trends of these accuracies per epoch, and it's noticeable that GAP converges faster but reaches a plateau early. We do see an unstable epoch 29 in the GAP training line, which drops down to around 3% accuracy, showing that the model was unstable in early-mid training over the GAP application. Top-5 accuracy for GAP crosses 79% at epoch 38 itself, whilst SOTAQlassifierHead plateaus after reaching the 77.7% peak.



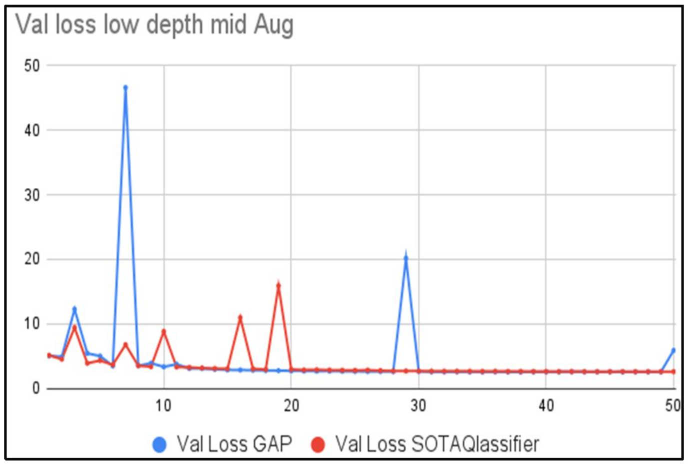


Unlike the validation accuracy curves, the training accuracy curves show a very clear GAP lead. The top 1 accuracy for GAP crosses 59% accuracy at epoch 41, while the Qlassifier peaks out at 55.35% after epoch 50. The top-5 accuracies have peaks of 82.58 at epoch 50 and 79.3 at epoch 58, respectively. GAP overall shows a very rapid convergence in its validation accuracy, just at the cost of instability. One thing to be noticed is that the training accuracies have a larger gap than the validation accuracies, thus already illustrating a case of inbuilt regularization in SOTAQlassifierHead over GAP, which doesn't regularize as much. This brings about a sense of SOTAQlassifierHead being better in generalized domains without overfitting, which was observed to a larger extent in GAP, even though the Qonstraint backbone regularizes a lot by itself.



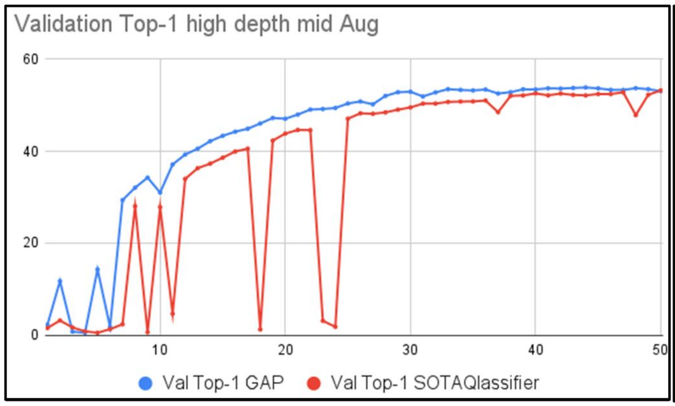


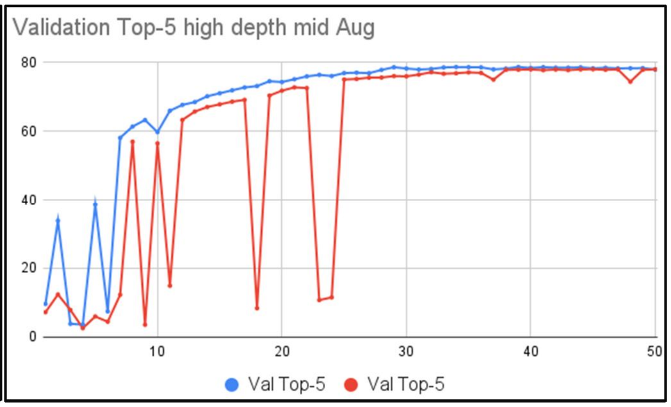
The validation loss curve shows the instability we noticed in the validation accuracy vividly. The GAP loss curve sees a late peak at epoch 29, whilst the other loss values are fairly similar, but SOTAQlassifierHead overall stays within a narrower range of validation losses, showing a bit more stable trend. This gives off a crucial inference. GAP achieved marginally higher validation and significantly higher training accuracies than SOTAQlassifierHead, but does it in a rather more volatile manner. This shows a case of trading off variance and bias. The Qlassifier deliberately gives up a fraction of accuracy for keeping the solution a lot more general, thus resulting in a performance that might be better on data that is out of distribution or in training cycles where instability can accumulate and ruin the model.



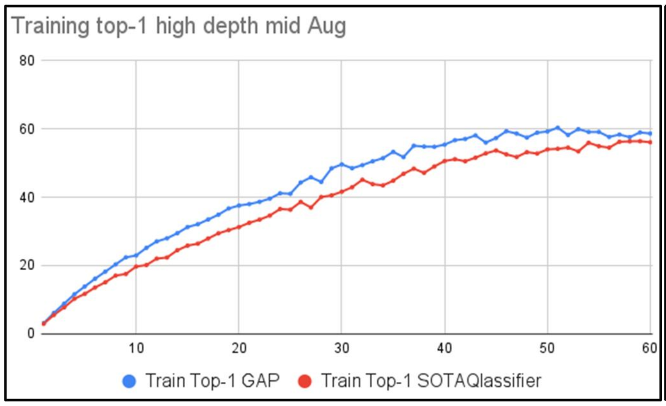
b) High depth

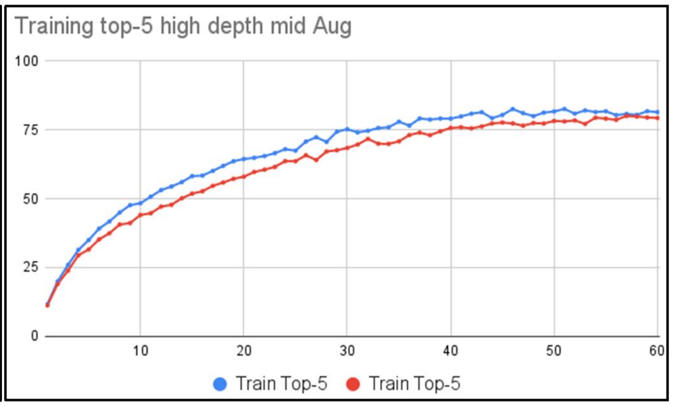
Looking at the validation accuracies in this deeper model environment, we see the SOTAQlassifierHead catching up to the GAP architecture's outputs. After both the heads cater to the model's early instability, GAP now peaks out at 53.88 percent accuracy at epoch 44 and stays around 53% itself for the second half of the training. SOTAQlassifierHead peaks out at 53.23% accuracy at epoch 50. The top-5 validation accuracy also shows a similar trend. GAP peaks at 78.72% accuracy and stays around 78% itself for the second half of the training, while the Qlassifier reaches 78.05% peak at epoch 45, which is just 0.02% higher than what it achieves on the epoch at which Val top-1 peaks.





Training accuracies still follow a similar trend to the lesser depths. GAP had a significant lead throughout the training accuracy, with it crossing 60% and 82.5% at epoch 51, whilst Qlassifier breaks 56% and 80% at epoch 58 for top-1 and top-5 respectively. This lead also explains the relatively more rapid convergence in validation accuracy, which is seen in GAP, which later on plateaus as a result of it learning the training data itself and not catching up to the patterns involved.

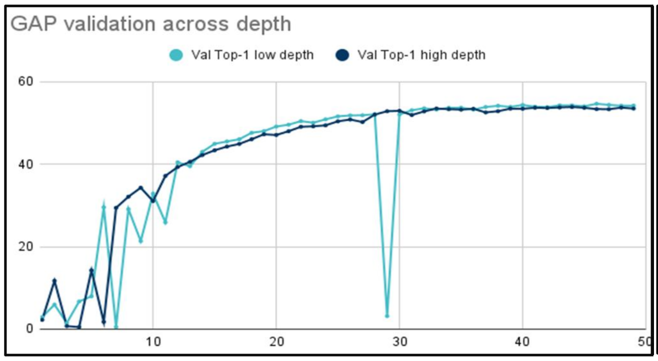


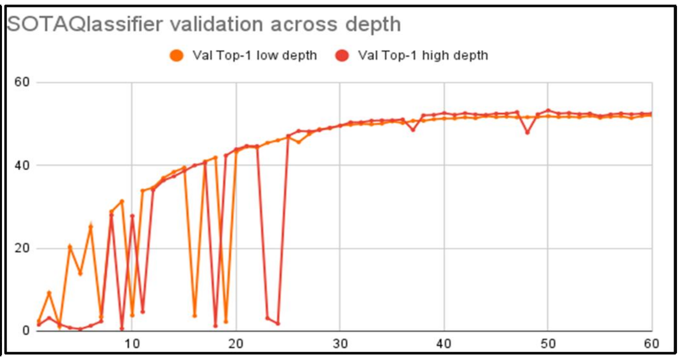


The fact that SOTAQlassifierHead and GAP both give similar results in this environment suggests that as the model grows in capacity, the advanced regularization and ensemble mechanisms become a lot more effective without generating any bias towards overfitting, as seen. The validation accuracy for both peaks is the same whole number for both top-1 and top-5, whilst the learning of training data shows a significant lead in GAP application, thus suggesting evident overfitting.

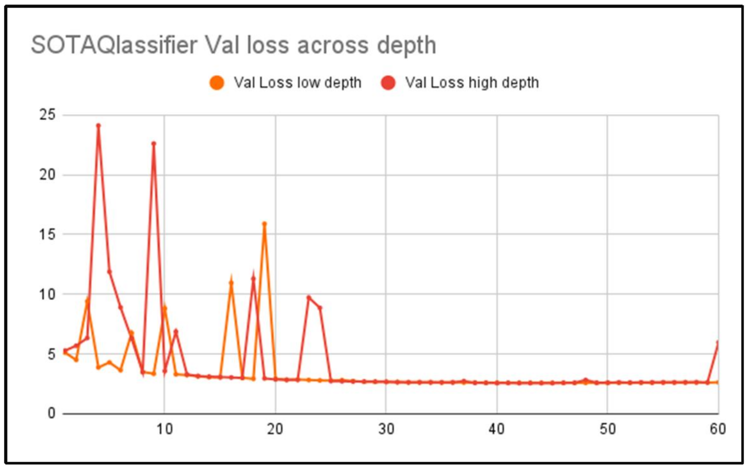
c) Comparison of depths

The above results show a very peculiar difference. GAP performed better under a lighter depth setting, whilst the SOTAQlassifierHead excelled at higher depths. The simpler and rather efficient design of GAP appears to be better matched to shallower models where representational demands are moderate. The ensemble and sophistication of SOTAQlassifierHead become advantageous once the model's capacity grows, and it can see deeper features.





One thing was common that whenever a head performed well, it was subject to heavy instability. The same has been plotted beneath. The higher depth on SOTAQlassifierHead showed many more and more pronounced loss peaks as compared to the lighter depth. This common nature of both the heads shows that whenever a model outputs better learning, volatility becomes a part of the process with Qonstraint.

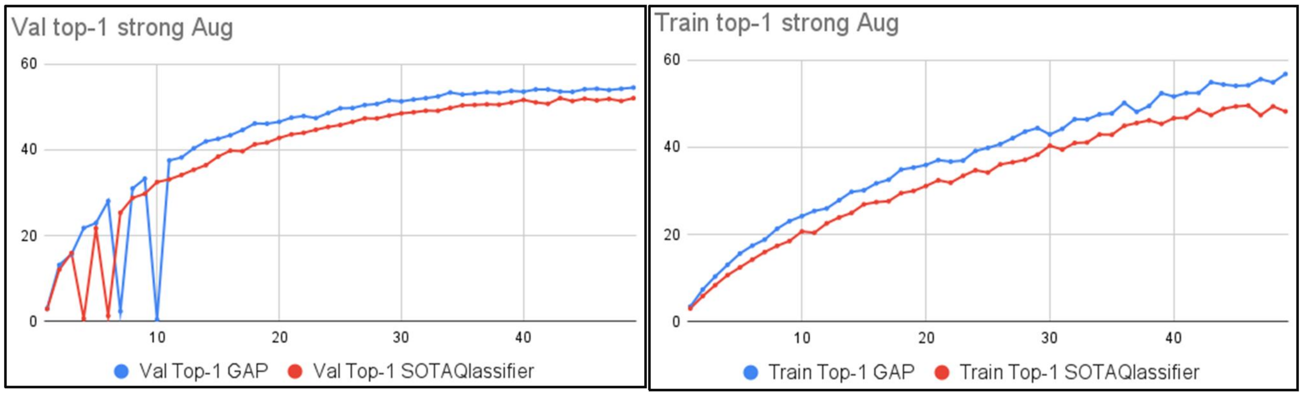


3) HIGH Augmentation

This augmentation regime was set very aggressively. A bit too aggressive for accuracy optimization. This section lets us see if Qonstraint 5.9 still has learning potential in extremely unrealistically augmented data. The phase-1 itself is strong and applies random crop down to a scale of 0.8, which later transcends up to 0.65. Color jittering is doubled from 0.2 to 0.4 in the two phases. Random erasing ranks up to 0.3, probably, which, in combination with the random cropping, lets the model be subjected to very limited information per image.

a) Low depth

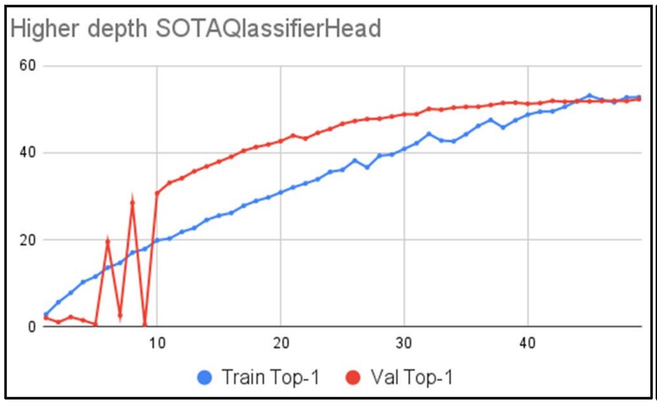
The depth was kept low, and both heads still showed promising results. SOTAQlassifierHead crossed 52% on the 43rd epoch, whilst GAP crossed 54.5% accuracy within 50 epochs. Thus showing that when global attention is given the responsibility to cater to classification, it's able to make up a global sense from the data and can predict accurately from even a small fraction of the image, and that too heavily augmented.

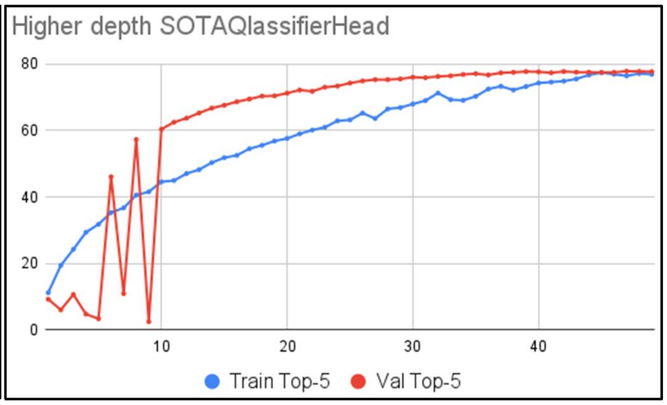


Though this is unrealistic, this shows the applicability of Qonstraint's architecture given the right choice of the classifier head based on the context and details of classification. The training accuracy explains this trend very well too, as both converge till the end of training, still showing model potential, but the finishing peak of SOTAQlassifierHead being just shy of 50% whilst GAP crossing 56%, thus implying the applicability of GAP in harsh augmentation regimes, which makes it so efficient in so many renowned models. The simplicity of GAP thus proved to be really effective even at low depths of the models, particularly performing better at the lower depth, taking advantage of its expressivity without the loss of vanishing gradients.

b) High depth

As we had observed earlier that SOTAQlassifierHead performed better under the higher depth setting, it was given a shot with higher depths too, and it still showed similar trends, this time learning up to its expectations. Now it had matching training and validation accuracies, thus suggesting that the shallower depths were leading to underfitting. This time, it reached an accuracy just shy of 53% showing the strengths of the novel approach of SOTAQlassifierHead.





4) Conclusion

This in-depth comparison of the two heads thus extends to explain the two classifier heads and the applicability of SOTAQlassifierHead in the future, with optimizations made. Being a novel idea and starting from scratch, it showed competitive results with the globally adopted head and showed matching validation accuracies with lower training accuracy in the medium augmentation, high depth, which becomes the key of this study. The GAP head demonstrates its remarkable efficiency, whilst the layered design of SOTAQlassifierHead comes with model capacity and data complexity, but much better regularization and a bias towards "learning" and not mugging up. As this study concludes, we see that neither approach is globally superior; GAP shows a slight boost in accuracy, which comes at its own cost. The relative strengths of the two can thus be hybridized in a way that brings out a better result overall. The continued pattern of the novel head performing better under higher depths shows that it's hungry for feature map richness, which was the intent with which Qonstraint was made. Later turns out TinyImageNet is a challenge that is based on extracting most of what's there, but with the added factor of preserving information, which GAP does the best. This leaves the door open for future research, which will enhance this classifier head battle into something productive.

V. DISCUSSION

The experimental results, novelty, and mosaic of techniques and approaches in Qonstraint 5.9 offer several insights into the design and architecture of hybrid vision models for small-scale datasets. This section aims to provide context for the model's innovations and its practical strengths, while also reflecting on its limitations and broader implications for vision AI research and deployment.

A. Why Qonstraint 5.9 Works

The effectiveness of Qonstraint 5.9 comes from the reliance on three innovations acting in coherence:

1) Hybrid Inductive Bias: Qonstraint can demonstrate how the integration of convolutional operations with transformer-style attention achieves an inductive bias that is particularly beneficial for data-constrained regimes. The convolutional stem, which extracts local spatial features, the neighborhood attention modules capturing the long-range dependencies, and the hybrid approach, all in all, enabling the model to generalize effectively without having any dependency on large-scale pretraining, which has acted as a critical advantage for domains where labelled data is rather scarce to obtain. This gives it applicability in real-life scenarios as well.
2) Confidence-Driven Dynamics: One feature that makes Qonstraint stand apart is its reinforcement learning inspired training control, which is adaptive as shown above. Controlling learning rate, momentum, and EMA decay in real time has proven to be fruitful. This accelerates convergence in early training by boosting the learning rate when any improvement is detected. Recovery after plateaus and focusing on harder samples is thus ensured, and adaptive regularization strengths are also an added benefit. These dynamics collectively bring about a self-supervised optimization setting, which brings about the sense of "learning how to learn" in the training process's regime itself, leading to a more optimal convergence.

3) Architectural Stability: Qonstraint has a mosaic-like classifier head, which updates its effects each epoch. As discussed in the paper, the head is designed to adaptively learn what to prioritize and when, creating a system that makes the model highly efficient at its classification task. The ensemble-like structure helps prevent overfitting and maintains consistent validation performance, even when learning rates and reloads vary.

B. Empirical impact of training Strategie

Various implementation choices collectively support the model's empirical success. The manual cosine restarts, which serve as a fallback to the optimized confidence learning mechanism, gradient clipping, dynamic augmentation, checkpoint reloading, and hardness-based loss weighting, all applied at different stages of learning, demonstrate how Qonstraint is highly effective in its function.

C. Limitations

Despite the strengths we observe, Qonstraint also has its limitations, listed as follows:

1) Depth Scalability: Adding any more than five QonvViT blocks will lead to gradients vanishing down the blocks. This happens because of cumulative normalization effects observed in residual connections. Some sort of gradient highway exploration is necessary to address this issue, which will make this model retain more of its gradients, thus enhancing the learning process.
2) Pretraining Dependency Trade-off: Qonstraint was always built in the regime of not utilizing any pretrained data. This enhances accessibility as intended, but comes with a trade-off, that being a constraint on domain transferability and the ultimate accuracy ceilings of the model, particularly when scaling to a larger dataset. Though this was never the purpose of beating with Qonstraint, it is still a limitation.
3) Hardware-Specific Optimization: The model's VRAM efficiency is tuned for GPU P100. INT8 quantization on ARM-based edge devices increases latency; thus, even though utilizing a publicly available GPU, it still indicates a need for hardware-aware architecture search and quantization settings.
4) Architectural gaps: Even though the research and explanation of this paper describe Qonstraint 5.9 in great depth, including everything it was based upon, it still has loopholes which have been left open and have been acknowledged in the paper too. Further efforts can upgrade this model to beat more benchmarks.

D. Broader Impact and Future Directions

Qonstraint's design is meant to support deployment in resource-constrained settings. Its efficiency in its tasks makes it a practical option to be deployed over edge AI applications. Real-time inference on hardware that's affordable enables applications in agriculture and portable healthcare, thus adding to the wave of making AI accessible to all. The low requirements make it well-suited for real-time applications.

Examples of the same can be real-time crop disease detection based on time-wise pictures of the crops no a particular time of the day, helping analyze the pictures in a similar setting. Food security has been on the cards as an initiative in food security. Another popular application can be facilitating point-of-care diagnostic analysis just by using smartphone cameras, so as to enable early detection in sections of society that aren't served well. Qonstraint also facilitates low-cost deployment of vision sensors in various other domains. The conservation efforts can be used in wildlife, pollution detection, disaster response, low-latency traffic detection, etc. Talking about model architecture, the text itself mentions points to be pondered upon for potential research in the future. The second phase of learning, which follows after the forced exploration boost, has not been given much attention and can be crucial for the model's performance. Another thing is how to mix the applications of the two classifier heads discussed into something fruitful, which makes the model reach its optimal classifier head, having the advantages of both, the globally accepted head and the novel head.

The absence of any pretraining barriers means that Qonstraint automatically lowers the entry barrier for anyone who lacks access to massive datasets, thus making SOTA-level vision AI more accessible to institutions and innovators in developing regions. The architecture is very adaptive, thus confidently being a viable option to specialize it in specific domains without the overhead of pretraining. The faster training observed and inference reduce energy consumption, thus contributing to environmentally sustainable AI research.

Applications of Qonstraint are endless. It can be extended with few-shot learning to enable rapid adaptation to new domains, thus working on reduced data. A research that can be followed and appended to this is using Qonstraint's modular design for automated neural network architecture search, having hardware constraints as parameters, and the development of efficient models.

VI. ACKNOWLEDGEMENT

I express my gratitude towards everyone who supported and helped me throughout this project. I hereby claim that this research was conducted as a part of my internship at Mindrops under the guidance and crucial inputs of the company's CEO, Rohit Gupta. I am also grateful to the faculty at BITS Pilani KK Birla Goa Campus who supervised me while working on this project. I also thank my friends and family who helped me tackle the issue of resource limits on Kaggle GPUs, and also their encouragement and abilities, which sustained my urge to keep making this research better.

REFERENCES

[1] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, Illia Polosukhin, "Attention is All You Need," Neural Information Processing Systems, 2017. https://arxiv.org/pdf/1706.03762.pdf
[2] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, Neil Houlsby, "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale," International Conference on Learning Representations, 2021. https://arxiv.org/pdf/2010.11929.pdf
[3] Li Yuan, Yunpeng Chen, Tao Wang, Weihao Yu, Yujun Shi, Zihang Jiang, Francis E.H. Tay, Jiashi Feng, Shuicheng Yan, "Tokens-to-Token ViT: Training Vision Transformers from Scratch on ImageNet," International Conference on Computer Vision, 2021. https://arxiv.org/pdf/2101.11986.pdf
[4] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, Hervé Jégou, "Training data-efficient image transformers & distillation through attention," International Conference on Machine Learning, 2021. https://arxiv.org/pdf/2012.12877.pdf
[5] Kai Wang, Yifan Sun, Jian Liang, Chunjing Xu, "Learning Correlation Structures for Vision Transformers," arXiv, 2023. https://arxiv.org/pdf/2404.03924.pdf
[6] Daehee Yun, Yong Man Ro, "SHViT: Single-Head Vision Transformer with Memory-Efficient Macro Design," arXiv, 2024. https://arxiv.org/pdf/2401.16456.pdf
[7] Ali Hassani, Steven Walton, Hessam Bagherinezhad, Mohammad Rastegari, "Neighborhood Attention Transformer," Conference on Computer Vision and Pattern Recognition, 2023. https://openaccess.thecvf.com/content/CVPR2023/papers/Hassani_Neighborhood_Attention_Transformer_CVPR_2023_paper.pdf
[8] Sachin Mehta, Mohammad Rastegari, "MobileViT: Light-weight, General-purpose, and Mobile-friendly Vision Transformer," International Conference on Learning Representations, 2022. https://arxiv.org/pdf/2110.02178.pdf
[9] Abhishek Abai, Rahul Rajmalwar, "DenseNet Models for Tiny ImageNet Classification," arXiv, 2019. https://arxiv.org/pdf/1904.10429.pdf
[10] Stanford CS231n, "Tiny ImageNet Challenge Report," 2017. http://cs231n.stanford.edu/reports/2017/pdfs/300.pdf
[11] Amir Irandoust, Hamed R. Tavakoli, Mohammad Sabokrou, "Training a Vision Transformer from Scratch in Less than 24 Hours with 1 GPU," Neural Information Processing Systems Workshop, 2022. https://arxiv.org/pdf/2211.05187.pdf
[12] Andreas Steiner, Alexander Kolesnikov, Xiaohua Zhai, Ross Wightman, Jakob Uszkoreit, Lucas Beyer, "How to Train Your ViT? Data, Augmentation, and Regularization in Vision Transformers," 2021. https://arxiv.org/pdf/2106.10270.pdf
[13] Yongming Rao, Wenliang Zhao, Benlin Liu, Jiwen Lu, Jie Zhou, "DynamicViT: Efficient Vision Transformers with Dynamic Token Sparsification," Neural Information Processing Systems, 2021. https://arxiv.org/pdf/2106.02034.pdf
[14] Zongwei Wang, Ioan A. Voiculescu, "Triple-View Feature Learning for Medical Image Segmentation," Medical Image Computing and Computer Assisted Intervention, 2022. https://arxiv.org/pdf/2208.06303.pdf
[15] "Cancer-Cell Deep-Learning Classification by Integrating Spatial, Temporal, and Quantitative Information," PubMed Central, 2021. https://pmc.ncbi.nlm.nih.gov/articles/PMC8699730/
[16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun, "Deep Residual Learning for Image Recognition," arXiv, 2015. https://arxiv.org/pdf/1512.03385.pdf
[17] Karen Simonyan, Andrew Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," arXiv, 2014. https://arxiv.org/pdf/1409.1556.pdf
[18] Forrest N. Iandola, Song Han, Matthew W. Moskewicz, Khalid Ashraf, William J. Dally, Kurt Keutzer, "SqueezeNet: AlexNet-level Accuracy with 50x Fewer Parameters and <0.5MB Model Size," arXiv, 2016. https://arxiv.org/pdf/1602.07360.pdf
[19] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, Andrew Rabinovich, "Going Deeper with Convolutions," arXiv, 2015. https://arxiv.org/pdf/1409.4842.pdf
[20] Ningning Ma, Xiangyu Zhang, Hai-Tao Zheng, Jian Sun, "ShuffleNet V2: Practical Guidelines for Efficient CNN Architecture Design," arXiv, 2018. https://arxiv.org/pdf/1807.11164.pdf
[21] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, Liang-Chieh Chen, "MobileNetV2: Inverted Residuals and Linear Bottlenecks," arXiv, 2018. https://arxiv.org/pdf/1801.04381.pdf
[22] Dongyoon Han, Jiwhan Kim, Junmo Kim, "Deep Pyramidal Residual Networks," arXiv, 2016. https://arxiv.org/pdf/1610.02915.pdf
[23] Sergey Zagoruyko, Nikos Komodakis, "Wide Residual Networks," arXiv, 2016. https://arxiv.org/pdf/1605.07146.pdf