Final Report: Is Efficient-CapsNet the solution to medical imaging challenges in developing countries?

G080 (s2450521, s2449242, s2416496)

Abstract

Current state-of-the-art for medical imaging, convolutional neural networks, require computationally intense training routines and very large datasets. A shortage of field experts to provide expert annotations and a lack of computing resources makes it difficult to adopt new methods in developing countries. (Hinton et al., 2011; Sabour et al., 2017) proposed capsule networks to overcome the limitations with existing CNN approaches. A recent implementation, Efficient-CapsNet, uses self-attention to enhance routing and uses an extremely low number of parameters. We believe applying the architecture in developing countries, where healthcare is least accessible, could help close the global healthcare inequality gap. We present the first investigation into the generalization of Efficient-CapsNet when challenges commonly found in medical datasets are imposed on training data; namely, number of samples is limited and augmentation is absent during training. Efficient-CapsNet significantly outperforms our low-parameter CNN baseline and ResNet18, in terms of classification accuracy and distributional generalization, in such a setting. Our implementation is available at https://github.com/ben-j-barlow/caps-net-project.

1. Introduction

Artificial intelligence (AI) advances promise brighter futures in many industries. In healthcare, computer vision techniques have become extremely common; publications of diagnostic imaging through AI has increased from 100-150 in 2007-2008 to 1000-1100 in 2017-2018 (Tan & Le, 2021). The benefits brought by AI could be most profound in developing countries, where healthcare is least accessible. However, current state-of-the-art is difficult to implement in such regions. We investigate if a new model for computer vision, Efficient-CapsNet (Mazzia et al., 2021), can provide a solution by assessing its ability to generalize when common healthcare dataset limitations are imposed.

Vision computing has been dominated by convolutional neural networks (CNNs) since they became state-of-the-art for image classification (Krizhevsky et al. 2012; He et al. 2016; Simonyan & Zisserman 2014), object detection (Ren et al., 2015) and instance segmentation (Chen et al. 2017; Ronneberger et al. 2015). Some examples of applications in healthcare are brain hemorrhage classification, interstitial lung disease classification, and breast cancer identification (Jnawali et al., 2018; Li et al., 2014; Spanhol et al., 2016). However, CNNs present limitations felt especially by developing countries. Their training is computationally intense and requires large training sets to achieve strong generalization. A scarcity of medical professionals to provide expert annotations; privacy issues in data collection; and a shortage of computing resources; all combine to make it challenging for developing countries to adopt current state-of-the-art.

The popularity of CNNs stems from their offering of translation invariant features: features seen at one spatial location during training can be detected at a different location during testing. This property is permitted by spatial reduction (pooling) layers, which also offer rotation invariant features over a small, bounded range (Hinton et al., 2011). Nonetheless, pooling layers are a limiting factor in performance because they discard the precise location and pose of an object's features (Hinton et al., 2011). This is significant in the context of the entire network because the spatial relationship between features, which can be essential to determining an object's class, is gradually lost with each downsampling layer.

Problems with current approaches inspired Hinton et al. (2011) to propose grouping neurons for vision computing into "capsules". Unlike activations that represent the presence of features in CNNs, capsules cooperate in synergy to represent different properties of the same entity. In capsule networks (Sabour et al., 2017), low-level capsules recognize object parts and then propagate information to high-level capsules which recognize wholes. High-level capsules consider the spatial relationship between parts when recognizing wholes, thereby overcoming the limitations of pooling layers in CNNs. For example, a capsule network will not recognize a face if atypical spatial permutations of mouth, nose, and eyes are found, whereas, a CNN could make that mistake (Sabour et al., 2017).

Six years after Hinton's proposal, Sabour et al. (2017) achieved near state-of-the-art performance on MNIST (LeCun et al., 1998) with the first implementation of capsules in a neural network. Jiménez-Sánchez et al. (2018) and Mobiny & Van Nguyen (2018) have since applied capsule networks to medical imaging tasks, but did not consider the ability to scale capsule networks in developing countries. A recent implementation, Efficient-CapsNet (Mazzia et al., 2021), offers a promising solution since its extremely bare architecture substantially reduces the need for computationally heavy training. It achieved state-of-the-art performance on MNIST with only 2% of the parameters of the Sabour et al. (2017)'s original capsule network implementation.

To determine if Efficient-CapsNet is suitable for addressing the medical imaging problem in developing countries, a number of milestones must be achieved in the research. Firstly, the ability of the network to generalize when training data is scarce must be examined. Secondly, since typical augmentation techniques are difficult to apply to medical datasets, an assessment of

Efficient-CapsNet's ability to outperform CNNs in the absence of augmentation must be conducted. Thirdly, a method for scaling the network to permit high performance on medical datasets (more complex than MNIST) must be found. Due to time constraints imposed by the project we focus only on tasks one and two and make a suggestion on how to achieve task three.

Specifically, we answer the following research questions:

1. How well do capsule networks generalize in a low-parameter-small-data[1]

2. Do capsule networks alleviate the need for data augmentation found with CNNs? [2] How does this change in a small-data setting?

We start by comparing a low-parameter capsule network (Efficient-CapsNet) to a low-parameter CNN baseline and evaluating generalization using two approaches. By varying the number of samples seen during training, we create a controlled small-data environment. We then train our networks using augmentation strategies and compare results to a popular CNN in the literature.

Finally, to the best of our knowledge, our key contributions are:

- The first assessment of capsule networks' performance in a low-parameter and small-data setting simultaneously.[3]

- The first set of ablation studies applied to Efficient-CapsNet.

- The first application of Efficient-CapsNet to a more complex dataset than MNIST (LeCun et al., 1998). [4]

2. Data set and task

We use the CIFAR10 dataset (Krizhevsky et al., 2009) for all experiments. The dataset contains 10 mutually exclusive classes (airplane, automobile, bird, cat, deer, dog, frog, horse, ship, and truck) and contains full variations in pose. There are 60000 images with 6000 per class for training and 10000 images with 1000 images per class for testing.

Additionally, to test the true generalization capability of our models, we evaluate on a new test set of CIFAR10 with 2000 images proposed by Recht et al. (2018; 2019). The authors argue that because CIFAR10's validation set has been widely benchmarked, prior models have unintentionally overfit to the original test set because performance has been optimized w.r.t. the original test set itself. The authors concluded that existing models should expect a drop-off in accuracy between 4% to 15% on their test set. For the later sections, we define the original test set as the validation set, and the new test set as test set.

The selection of CIFAR10 was largely because the class objects and their variations through affine transformations, noise, natural background scene, and the presence of RGB colour channels presents a reasonable level of complexity. Importantly, CIFAR10 is more complex than MNIST (LeCun et al., 1998) which can easily be classified with low level features such as edges and blobs. We hypothesize this challenge will enable a capsule network's superior ability to generalize in a low-parameter-small-data setting to become apparent. Furthermore, Efficient-CapsNet is yet to be applied to CIFAR10, thereby promising our work is a novel extension of theirs.

Nonetheless, the images are of dimension 32×32 with only 10 classes which is still some distance from images found in the real-world in terms of complexity (Krizhevsky et al., 2009). We suggest that applications to complex medical datasets should only be performed after research directions outlined in Sec. 1 are pursued and Efficient-CapsNet is proved scalable.

Whilst capsule networks have been used in object detection (Pan & Velipasalar, 2021) and image segmentation tasks (Koresh et al., 2021), image classification is a natural task choice here since it permits direct comparison with the literature we are extending: Mazzia et al. (2021). We evaluate on classification accuracy and distributional generalization (Nakkiran & Bansal, 2020), as explained in 5.

Other details to note regarding data set are our sampling and pre-processing techniques. The term "small-data" is used to describe training data with ≤ 600 samples per class throughout this paper. To test all models in a small-data setting, we run experiments on 10%, 5%, and 1% of the training dataset (600, 300, and 60 samples per class). Sampling without replacement prevents duplicate images and creates disjoint training sets for each training run, whilst stratification ensures original class distribution is preserved. Dataset-wise normalization is performed on each channel independently to prevent data leakage. Normalization is performed after data augmentation; further detail is provided in Sec. 5.

It is necessary to address the modification to our task made since coursework 3. We initially planned to present two investigations: one in a fully-supervised, small data setting, and another in a few-shot setting. Hence, coursework 3 introduced miniImageNet (Vinyals et al., 2016) for few-shot learning. We instead focus our second investigation on ablation studies in a fully supervised, small-data setting, since it allows consistent use of the term "small-data" throughout our experiments.

3. Related work

As outlined in Sec. 1, Hinton et al. (2011) first proposed capsules. In contrast to the scalar output of convolution and downsampling layers, capsules output a vector containing highly informative values. The length (norm) of the output vector represents the probability the entity encoded by the capsule is present over the limited domain characterized by the capsule. Meanwhile, the vector's orientation represents pose relative to the implicitly defined canonical version of that entity. Each dimension in the output vector implicitly captures a property of the entity, such as thickness, skew, and stretch, as demonstrated by Fig. 7. The term "pose" is used to encapsulate all of these properties.

[1]Definitions for "low-parameter" and "small-data" are provided in Tab. 1 and Sec. 2.

[2]A modification to this research question has been made since coursework 3. See Sec. 2 for an explanation.

[3]Both have been done independently. For example, see Mazzia et al. (2021) and Jiménez-Sánchez et al. (2018).

[4]See 2 for an explanation on what makes CIFAR10 a more complex dataset as compared to MNIST.

Sabour et al. (2017) achieved near state-of-the-art performance on MNIST with "CapsNet"; the first neural network comprised of capsules. Whilst the 0.25% test error in Sabour et al. (2017) was inferior to state-of-the-art at the time (0.21%; Wan et al. 2013), their contribution was significant because their single model without use of rotation and scaling during training was far superior to Wan et al. (2013)'s 0.39% test error achieved without data augmentation. This was an early indication capsule networks could overcome a CNNs lack of ability to generalize without sufficient augmentation.

By Sabour et al. (2017)'s own admission, there exist many methods to implement the general idea of capsules. Improvements to their "routing-by-agreement" mechanism, named "dynamic routing", has been the main focus in literature (Lin et al., 2018; Hinton et al., 2018). Choices made on routing, the process by which information gets propagated from a layer of child capsules to a layer of parent capsules, dictates the number of parameters in the model and can be more influential on performance than conventional hyperparameters such as batch size, momentum, and learning rate (Lin et al., 2018; Patrick et al., 2022). Routing improvements have been made in terms of speed of convergence (Ramasinghe et al., 2018), performance of the trained network (Zhao et al., 2019), and the required number of parameters (Hinton et al. 2018; Ren et al. 2019; Mobiny & Van Nguyen 2018).

Mazzia et al. (2021) was the first contribution to propose a solution that exhibited the full potential of capsule networks. Efficient-CapsNet's extremely bare architecture and highly parallelizable self-attention routing mechanism allowed them to demonstrate that continuing to train capsule networks with a large number of parameters hides their intrinsic capability to generalize to novel viewpoints. They achieved state-of-the-art on MNIST (LeCun et al., 1998) with 161K parameters (a fraction of CapsNet's 6.8M), which was especially impressive since other state-of-the-art models for MNIST have over 1M (Byerly et al. 2021: 1.5M, Assiri 2020: 1.4M).

Applications of capsule networks to medical imaging tasks have been performed too. Interestingly, these investigations also suggested capsule networks as a potential solution when training data is limited. Jiménez-Sánchez et al. (2018) applied CapsNet to medical datasets TUPAC16 (Veta et al., 2019) and DIARETDB1 (Kauppi et al., 2007) and Mobiny & Van Nguyen (2018) performed an experiment on Computed Tomography (CT) chest scans. They both concluded CapsNet is favourable to CNNs (applied to their data), but significantly outperforms on small size datasets. However, neither of these works considered a very low-parameter solution required by developing countries. There ceases to exist an investigation that compares capsule networks and CNNs in a small-data and extremely low-parameter setting simultaneously.

State-of-the-art performance for our task and data is not considered relevant due to the nature of our investigation. Current state-of-the-art for image classification on CIFAR10 leverage huge amounts of parameters (Dosovitskiy et al. 2020: 632M; Wu et al. 2021: 277M) and undergo pre-training (Tan & Le, 2021; Kolesnikov et al., 2020) with complicated datasets such as ImageNet (Deng et al., 2009). We instead focus on our low-parameter CNN baseline for comparison.

4. Methodology

We define our general implementation of capsule networks in 4.1. All definitions here are consistent in CapsNet and Efficient-CapsNet. We then describe our specific implementations: CapsNet (Sabour et al., 2017) in Sec. 4.2 and Efficient-CapsNet in Sec. 4.3 (Mazzia et al., 2021). Additional information about our methodology is provided in subsequent sections.

4.1. Capsule network: in general

Convolutional layers in CNNs are effective feature detectors: it is important to retain their power when exploiting the endowment of capsules. Initially, a $W \times H \times C$ input sample $\boldsymbol{X}^{(0)}$ passes through a downsampling-free CNN comprising an arbitrary number N_{Conv} layers. The l-th layer's output $\boldsymbol{X}^{(l)}$,

$$\boldsymbol{X}^{(l)} = \mathrm{ReLU}\Big(\mathrm{Conv}_{\{k^{(l)},s^{(l)},f^{(l)}\}}(\boldsymbol{X}^{(l-1)})\Big), \tag{1}$$

depends on a choice of kernel size $k^{(l)}$, stride $s^{(l)}$, and number of feature maps $f^{(l)}$. The CNN eventually outputs a mapping of input $\boldsymbol{X}^{(0)}$ to a higher-dimensional space. Using (1), the $W^* \times H^* \times C^*$ output is $\boldsymbol{X}^{(N_{Conv})}$ which we rename to $\boldsymbol{X}^{(CNN)}$ for simplicity.

The remainder of both our networks is comprised of capsules. In the general capsule network framework, the first capsule layer is followed by an unspecified number of capsule layers. As demonstrated by Fig. 1, only one subsequent layer of capsules is implemented in our architecture due to the relative simplicity of CIFAR10. In the first layer, named "PrimaryCaps", pixel intensities are converted into a vectorial representation to create N_p capsule outputs $\boldsymbol{u}_i$ with dimension d_p. This vectorial representation is maintained in the next layer, "ClassCaps", which outputs N_c vectors $\boldsymbol{v}_j$ with dimension d_c. Dimensionality is consistent across both our networks, with $d_p = 8$ and $d_c = 16$. The increase in dimensionality is justified by the fact higher level capsules capture more complex entities with more degrees of freedom, thereby requiring more dimensions to accurately represent entities as we ascend the hierarchy (Sabour et al., 2017).

The propagation of data from PrimaryCaps to ClassCaps gives rise to the first key difference between a capsule network and a traditional CNN or feedforward network. The output $\boldsymbol{u}_i$ of the i-th child capsule in PrimaryCaps is used to compute a prediction $\hat{\boldsymbol{u}}_{j|i}$ of the output $\boldsymbol{v}_j$ of the j-th parent capsule in ClassCaps. Specifically,

$$\hat{\boldsymbol{u}}_{j|i} = \boldsymbol{W}_{ij}\boldsymbol{u}_i, \quad (i = 1, \ldots, N_P, j = 1, \ldots, N_C) \tag{2}$$

where $\boldsymbol{W}_{ij}$ has dimensions $d_c \times d_p = 16 \times 8$ to project $\boldsymbol{u}_i$ to a space with dimension $\dim(\hat{\boldsymbol{u}}_{j|i}) = d_c = 16$. Glorot initialization (Glorot & Bengio, 2010) is adopted for $\boldsymbol{W}_{ij}$ before it is learned during training. The routing mechanism employed by ClassCaps takes $\hat{\boldsymbol{u}}_{j|i}$ as input and produces $\boldsymbol{v}_j$ as output. As devised in 3, the length (norm) of a capsules output represent the probability the entity represented by the capsule is present. Hence, computing $\|\boldsymbol{v}_j\|$, $j = 1, \ldots, 10$, enables a classification to be obtained. The final output is not anymore represented by a scalar $k \in \{1, \ldots, K\}$, but a vector $\boldsymbol{v}_{j^*}$ that captures the pose of the entity represented by the chosen class capsule $\boldsymbol{v}_{j^*}$, $j^* \in \{1, \ldots, K\}$.

The three components of the network in Fig. 1 are CNN, PrimaryCaps, and ClassCaps. Details of PrimaryCaps and ClassCaps behaviour for each model is explained in 4.2 and 4.3. For an understanding of the CNN component of each, see Figs. 5 and 6.

4.2. CapsNet

4.2.1. PrimaryCaps

Given the 32×32 input $\boldsymbol{X}^{(0)}$ and CapsNet's CNN implementation (see Fig. 5), PrimaryCaps receives an input tensor $\boldsymbol{X}^{(CNN)}$ with dimension $W^* \times H^* \times C^* = 24 \times 24 \times 256$. In PrimaryCaps, a convolution operation as defined in (1) is applied $(k = 9, s = 2, f = 256)$ to produce an $8 \times 8 \times 256$ tensor. Since CapsNet has 32 primary capsules at each location, the tensor is reshaped first to an $8 \times 8 \times 32 \times 8$ tensor, and then to a 2048×8 tensor $\boldsymbol{U}$ to produce $N_p = 2048$ capsules with dimension $d_p = 8$.

We have two strong requirements on the output of each capsule. Firstly, the length of a capsule's output must represent the probability the entity embodied by the capsule is present in the image. Secondly, the vector's orientation must be consistent to guarantee it represents the entity's pose. Hence, we define the output of PrimaryCaps as $\boldsymbol{u}_i = \text{squash}_{CN}(\boldsymbol{U}_{(i,:)})$, where $\boldsymbol{U}_{(i,:)}$ is the i-th row of $\boldsymbol{U}$, and squash_{CN} is defined as

$$\text{squash}_{CN}(\boldsymbol{a}) = \frac{||\boldsymbol{a}_i||^2}{1 + ||\boldsymbol{a}_i||^2} \frac{\boldsymbol{a}_i}{||\boldsymbol{a}_i||}, \tag{3}$$

where $\boldsymbol{a}$ is some vector of arbitrary length. Clearly, we obtain 2048 output vectors $\boldsymbol{u}_i$ that satisfy our conditions.

4.2.2. ClassCaps

ClassCaps receives prediction vectors $\hat{\boldsymbol{u}}_{j|i}$ as input, as defined in (2) $(i = 1, \ldots, 2048, j = 1, \ldots, 10)$. Each capsule within the layer receives a weighted sum over all prediction vectors

$$\boldsymbol{s}_j = \sum_i^{N_p} c_{ij} \hat{\boldsymbol{u}}_{j|i}, \quad (j = 1, \ldots, N_c) \tag{4}$$

where coupling coefficients c_{ij} control the contribution of the i-th capsule in PrimaryCaps to the j-th capsule in ClassCaps. Finally, the output of the j-th capsule is

$$\boldsymbol{v}_j = \text{squash}_{CN}(\boldsymbol{s}_j) \quad (j = 1, \ldots, 10).$$

Clearly, the squashing function imposes the same constraints on $\boldsymbol{v}_j$ as it did on $\boldsymbol{u}_i$ in Sec. 4.2.1. We then classify input $\boldsymbol{X}^{(0)}$ by adopting the approach explained in Sec 4.1.

4.2.3. Dynamic routing

Dynamic routing (Sabour et al., 2017) is the process by which CapsNet learns to propagate information from PrimaryCaps to ClassCaps. At inference time, the contribution of capsule i in PrimaryCaps to capsule j in ClassCaps, c_{ij} in (4), is known for all i, j. However, prior to training, the coupling coefficients are yet to be determined.

Initial logits b_{ij} are introduced to represent the log prior probabilities that capsule i should be coupled to capsule j. A softmax is used when computing c_{ij},

$$c_{ij} = \frac{\exp(b_{ij})}{\sum_k \exp(b_{ik})}, \tag{5}$$

to ensure $\sum_j c_{ij} = 1$. Initialising priors $b_{ij} = 0$ clearly sets coupling coefficients equal at the start of training, which we found to be sufficient for CIFAR10. The logits are iteratively refined by incrementing their value by the dot-product agreement $a_{ij} = \boldsymbol{v}_j \cdot \hat{\boldsymbol{u}}_{j|i}$ between the j-th capsule's output $\boldsymbol{v}_j$ and it's predicted outputs $\hat{\boldsymbol{u}}_{j|i}$ produced by each child capsule $i = 1, \ldots, 2048$. Of course, the greater the contribution of $\hat{\boldsymbol{u}}_{j|i}$ to $\boldsymbol{v}_j$, the greater the agreement a_{ij}. Like b_{ij}, the agreement a_{ij} is treated like a log-likelihood and is added to the initial logit b_{ij} in each iteration.

Algorithm 1 Dynamic routing (Sabour et al., 2017)

```
**Input:** Capsule prediction $\hat{\boldsymbol{u}}_{j|i}$, routing iteration $r$
$b_{ij} \leftarrow 0 \quad (i = 1, \ldots, N_p, j = 1, \ldots, N_c)$
for $l = 1$ to $r$ do
    $c_j \leftarrow \frac{\exp(b_{ij})}{\sum_k \exp(b_{ik})} \quad (j = 1, \ldots, N_c)$
    $\boldsymbol{s}_j \leftarrow \sum_i c_{ij} \hat{\boldsymbol{u}}_{j|i} \quad (j = 1, \ldots, N_c)$
    $\boldsymbol{v}_j \leftarrow \text{squash}_{CN}(\boldsymbol{s}_j) \quad (j = 1, \ldots, N_c)$
    if $i < r$ then
        $b_{ij} \leftarrow b_{ij} + \hat{\boldsymbol{u}}_{j|i} \cdot \boldsymbol{v}_j \quad (i = 1, \ldots, N_p, j = 1, \ldots, N_c)$
    end if
end for
return $\boldsymbol{v}_j$
```

4.3. Efficient-CapsNet

4.3.1. PrimaryCaps

Alike CapsNet, the main responsibility of PrimaryCaps is converting pixel intensities to a set of capsule vectors to be propagated to higher level capsules. However, choices made with regards to PrimaryCaps' convolution operation heavily reduces the number of parameters required for the capsule-creating process, compared to CapsNet. Furthermore, an alternative squashing function is used to boost the gradient during training.

Due to choices made in the network's CNN component (see Fig 6) and CIFAR10's image dimensions, PrimaryCaps receives an input tensor $\boldsymbol{X}^{(CNN)}$ with dimensions $11 \times 11 \times 128$. Instead of a traditional convolution operation, the input tensor undergoes depthwise separable convolution (Chollet, 2017). This process comprises two parts: a depthwise convolution (spatial convolution acting on each channel independently) followed by a pointwise convolution (a 1×1 convolution). Remarkably, the input image is reduced to a $1 \times 1 \times 128$ tensor by choice of a 11×11 kernel and stride 1 for depthwise convolution. Following this, the pointwise operation projects the tensor onto a new channel space with the number of channels chosen to be 128 again.

The resulting $1 \times 1 \times 128$ tensor is reshaped to a 16×8 matrix $\boldsymbol{U}$. As in CapsNet, a squashing function is applied to produce vector outputs with our desired properties. Mazzia et al. (2021) instead use

$$\text{squash}_{ECN}(\boldsymbol{a}) = \left(1 - \frac{1}{exp(||\boldsymbol{a}_i||)}\right) \frac{\boldsymbol{a}_i}{||\boldsymbol{a}_i||}, \tag{6}$$

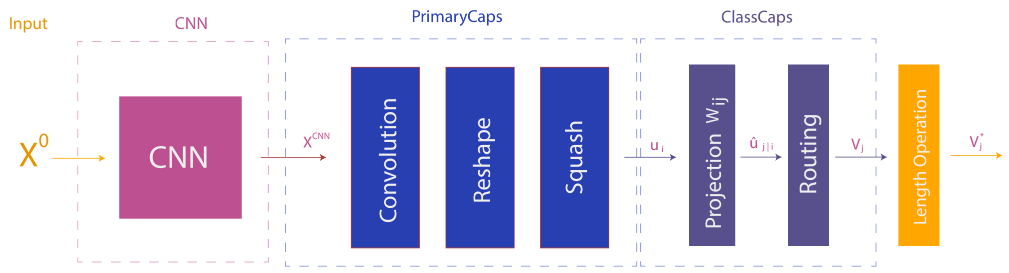


Figure 1. The model architecture followed by CapsNet and Efficient-CapsNet. The projection in ClassCaps and Length Operation are consistent across both models. But the CNN, PrimaryCaps and Routing components are implemented differently. Further details on their implementation for each model are provided in the appendix.

where $\boldsymbol{a}$ is an arbitrary vector, to produce $\boldsymbol{u}_i = \text{squash}(\boldsymbol{U}_{(i,:)})$, where $\boldsymbol{U}_{(i,:)}$ is the i-th row of $\boldsymbol{U}$. (6) is far more sensitive to gradient changes around 0 than (3), thereby decreasing the probability of gradients approaching 0 during training (Xi et al., 2017).

4.3.2. ClassCaps and Self-attention routing

For our definition of CapsNet, ClassCaps and dynamic routing had independent sections 4.2.2 and 4.2.3. This was necessary to provide a sufficient explanation regarding the computation of the coupling coefficients c_{ij}. However, in Efficient-CapsNet, the coupling coefficients are calculated directly (in a single pass) and the log priors are learned discriminatively alongside all other learnable parameters. Hence, the self-attention routing is considered to be entirely contained by ClassCaps, thereby omitting the need for a separate definition.

Define a $N_p \times N_c \times d_c = 16 \times 10 \times 16$ tensor $\hat{\boldsymbol{U}}$ using each prediction $\hat{\boldsymbol{u}}_{j|i}$

$$\hat{\boldsymbol{U}} = \begin{pmatrix} \hat{\boldsymbol{u}}_{1|1} & \dots & \hat{\boldsymbol{u}}_{N_c|1} \\ \vdots & & \vdots \\ \hat{\boldsymbol{u}}_{1|N_p} & \dots & \hat{\boldsymbol{u}}_{N_c|N_p} \end{pmatrix},$$

where $\hat{\boldsymbol{u}}_{j|i}$ is defined as in (2). Then, we define

$$\boldsymbol{A}_{(:,:,j)} = \frac{\hat{\boldsymbol{U}}_{(:,j,:)} \times \hat{\boldsymbol{U}}^T_{(:,j,:)}}{\sqrt{d_p}}, \quad (j = 1, \dots, N_c)$$

where $\boldsymbol{A}_{(:,:,j)}$ is a $N_p \times N_p \times 1 = 16 \times 16 \times 1$ tensor, $\sqrt{d_p}$ is used to stabilize training, and, by considering $j = 1, \dots, N_c$, we implicitly define the $N_p \times N_p \times N_c = 16 \times 16 \times 10$ self-attention tensor $\boldsymbol{A}$. For clarity, $\hat{\boldsymbol{U}}_{(:,j,:)}$ is $N_p \times 1 \times d_c$ and its transpose $\hat{\boldsymbol{U}}^T_{(:,j,:)}$ is $d_c \times 1 \times N_P$. The scalar $A_{(k,l,j)}$ contains the score agreement for the k- and l-th primary capsules' predictions for the j-th class capsule's output $\boldsymbol{v}_j$. Explicitly, this is the agreement of $\hat{\boldsymbol{u}}_{j|k}$ and $\hat{\boldsymbol{u}}_{j|l}$.

This permits the $N_p \times N_c$ coupling coefficient matrix $\boldsymbol{C}$ to be computed, which is defined by

$$\boldsymbol{C}_{(:,j)} = \text{softmax}\Big(\boldsymbol{A}_{(:,i,j)}\Big) \quad (j = 1, \dots, N_c).$$

Similarly to (5), the softmax is used to ensure the total contribution received by the j-th class capsule is 1, precisely $\sum_j c_{ij} = 1$. Next, the vector $\boldsymbol{s}_j$ is computed as

$$\boldsymbol{s}_j = \hat{\boldsymbol{U}}^T_{(:,j,:)} \odot \Big(\boldsymbol{C}_{(:,j)} + \boldsymbol{B}_{(:,j)}\Big) \quad (j = 1, \dots, N_C)$$

where $\boldsymbol{B}$ is the $N_p \times N_c$ log prior matrix containing all biases learned during training. Like CapsNet, initializing the log prior matrix as the zero matrix was sufficient for training on CIFAR10. Finally, $\boldsymbol{v}_j = \text{squash}_{ECN}(\boldsymbol{s}_j)$ is outputted from the j-th capsule $(j = 1, \dots, N_c)$ and an image classification can be obtained as explained in Sec. 4.2.

4.3.3. The background of self-attention routing

Approximate equivalence can be drawn between self-attention routing and Equation (1) in Vaswani et al. (2017), which is repeated here for convenience:

$$\text{Attention}(Q, K, V) = \text{softmax}\Big(\frac{QK^T}{\sqrt{d_k}}\Big)V.$$

In the self-attention routing mechanism, the query Q, key K, and value V take value approximately $\hat{\boldsymbol{U}}_{(:,j,:)}$. Direct equivalence is not possible since a log prior is added to the softmax's output before the dot product with V is performed.

4.4. Baseline

We sought a high performing low-parameter CNN baseline that provided to provide a fair comparison with Efficient-CapsNet. A near identical architecture, with a CNN, Primary Caps, and ClassCaps component was achieved. The CNN component was the same as in Efficient-CapsNet. PrimaryCaps utilised the same depthwise separable convolution, but introduced two fully-connected layers with 328 and 192 neurons, respectively. Finally, we add a 10 channel fully-connected layer for the classifier, which uses dropout (Srivastava et al., 2014) with probability $p = 50\%$, similar to the original baseline proposed in (Sabour et al., 2017). Adopting a near identical structure allows us to adequately evaluate the positive effects a capsule's vector representation brings over scalar representations found in CNNs.

4.5. Margin loss

As defined in Sabour et al. (2017), we use a margin loss

$$L_j = T_j \max(0, m^+ - \|\boldsymbol{v}_j\|)^2 + \lambda(1 - T_j) \max(0, \|\boldsymbol{v}_j\| - m^-)^2$$

during training. We compute L_j for each class capsule

$j = 1, \ldots, N_c$. If the entity represented by the j-th capsule is present in the image, $T_j = 1$; otherwise $T_j = 0$. We use Sabour et al. (2017)'s recommended values $m^+ = 0.9$ and $m^- = 0.1$ which we found to be sufficient for convergence on CIFAR10.

The authors argue that margin loss is used to allow the top-level capsule have a high-length vector only when the object within the class capsules appears in the images. λ reduces the magnitude of shrinking when the object is not present during training, while m^+ and m^- define the margins of the positive and negative classes, respectively.

5. Experiments

We found appropriate hyperparameters for all experiments manually due to the absence of configurations presented in Sabour et al. (2017) and Mazzia et al. (2021). We implement our models based on the TensorFlow (Abadi et al., 2015) implementation of both CapsNet and Efficient-CapsNet from (Mazzia et al., 2021) [5] which we customized for our experiments. We use the existing implementation of ResNet18 directly (Wood et al., 2022) and avoid using the pre-trained weights to ensure fair comparison. We trained our models (ResNet18, CapsNet, and Efficient-CapsNet) with AdamW optimizer (Loshchilov & Hutter, 2019) over 250 epochs, batch size $b = 128$, base learning rate $\gamma = 10^{-4}$ which we scaled linearly by $b\frac{\gamma}{16}$, and weight decay 10^{-7}. We schedule our learning rate with cosine annealing (Loshchilov & Hutter, 2017) without resets across 250 epochs. For the CNNs, we optimize the models by using standard cross-entropy loss while the CapsNets are optimized by margin loss as stated in 4.5. Additionally, we checkpoint our models based on the lowest validation loss throughout training. We chose this configuration since we found it optimal based on initial convergence over 100 epochs. See an explanation of our model choices in Tbl. 1 and model definitions in 4.

5.1. Efficient-CapsNet and classification accuracy

The aim of this experiment is to assess whether capsule networks generalize better than CNNs in a low-parameter-small-data setting. We impose common medical dataset challenges on training data by sampling three training sets (10%, 5%, and 1%) and train on the entire training set to provide a benchmark. Disjoint validation and test sets are used for evaluation. Each model is run 5 times to estimate the uncertainty of the result.

Given our research question specifies "small-data" and "low-parameter", we focus our attention to our baseline and ECN in Fig 2 (and 2 in the appendix). Firstly, the superior performance of our baseline over ResNet18 demonstrates we achieved our goal of producing a powerful baseline. This is vital for meaningful conclusions since comparison with a weak baseline would provide little information on the efficiency of ECN.

The slope of the baseline and ECN curves for sample size $\leq 10\%$, it is clear performance of Efficient-CapsNet degrades at a much slower rate than the CNN baseline as data decreases. The model's difference between classification accuracy on the validation set starts at 0.83% for 100% of the data, and increases to 3.19%, 3.93%, and finally 8.39% as data is decreased to 10%, 5%, and 1% (see 2). The gap of 8.39% on the validation set and 6.85% on test set for the most extreme data setting shows strong evidence capsule networks generalize better than CNNs in a low-parameter-small-data setting.

[5]Efficient-CapsNet original repository: https://github.com/EscVM/Efficient-CapsNet

Parameters	Model	
	CNN	Capsule network
High	ResNet18 (11.2M)	CapsNet (8M)
Low	Our baseline (200K)	Efficient-CapsNet (168K)

Table 1. The "parameter-model square" highlights the number of parameters in each network. By analysing common network sizes present in the literature, we define a "low-parameter" and "high-parameter" model to have $< 1M$ and $> 5M$ parameters, respectively. ResNet18's (He et al., 2016) popularity in the literature provides a well established high-parameter CNN baseline. To accurately assess the capability of capsules in a low-parameter setting, we require the high performing low-parameter CNN baseline explained in Sec. 4.4. Finally, Sabour et al. (2017)'s CapsNet was trained to provide a high-parameter capsule network baseline, thereby allowing comparison of Efficient-CapsNet along both axes of the "parameter-model square". For specific choices regarding definition of our baseline, CapsNet, and Efficient-CapsNet, see Sec. 4.

Despite the application of self-attention routing in the limited data setting being absent in the literature, there is indirect evidence to support our result. Jiménez-Sánchez et al. (2018) and Mobiny & Van Nguyen (2018) performed similar experiments with decreasing data available and concluded the gap between capsule networks and CNNs is most profound when maximal limitations are placed on training dataset size. Their results obtained with CapsNet[6] are still applicable to ECN since their fundamental properties are the same: capture an entity's entire pose in instantiation vectors rather than just detecting the presence of the entity.

5.2. Efficient-CapsNet and data augmentation

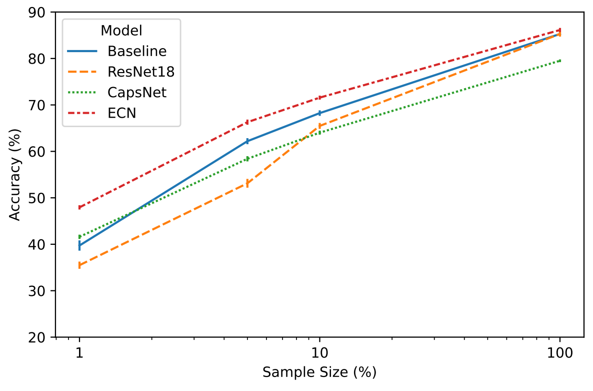
For assessing Efficient-CapsNet's need for data augmentation to generalize well, we use ResNet18 as a benchmark since this experiment does not demand a low-parameter baseline and ResNet18 is well established in the literature.

To consider Efficient-CapsNet's need for augmentation in a small data regime, we must first benchmark on the full training set. Fig 3 (and 3 in the appendix) shows both the CNN and capsule network benefit equally from a change from no augmentation to full augmentation, with validation set performance increasing by 10.74% and 10.41% to 85.55% and 86.34% for ResNet18 and ECN, respectively. The achievement of similar accuracy in all augmentation settings contradicts previous results that suggested capsule networks without data augmentation can perform better than CNNs with augmentation (Jiménez-Sánchez et al., 2018). We believe this inconsistency can be explained by their choice of a weak baseline (which is outperformed by LeNet (LeCun et al., 1998) in their experiment) and our choice of a strong benchmark: ResNet18[7].

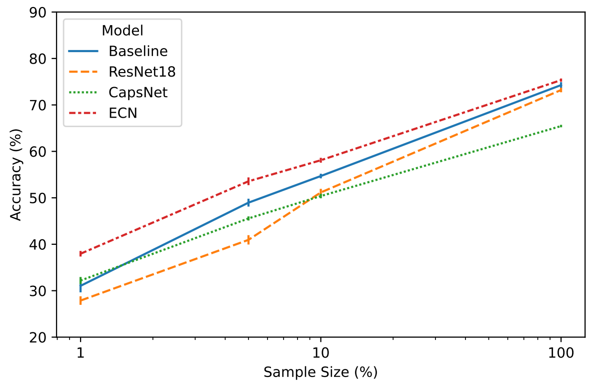
In the small-data setting, Efficient-CapsNet significantly out-

[6]We also implement CapsNet and observe it outperforms all other models in the most extreme setting; see 2.

[7]ResNet18 is stronger due to a significant increase in parameters.



(i) Validation set

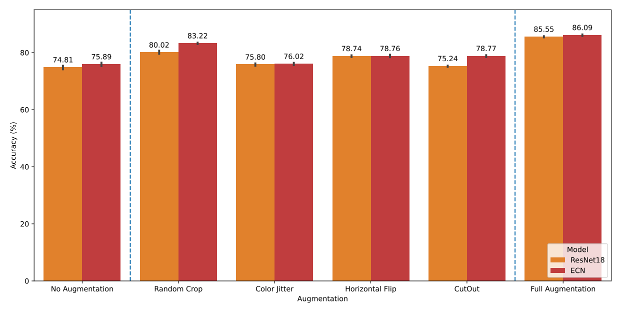


(ii) Test set

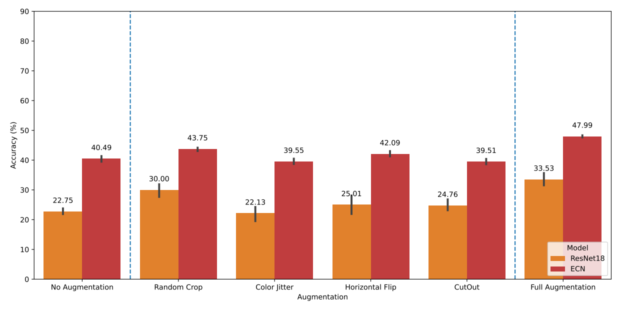
Figure 2. Classification accuracy of our trained models as dataset size is increased.

performs ResNet18. This result is not new; we already saw this in Fig. 2. However, the focus here is how much each network depends on augmentation. Moving from full augmentation to no augmentation causes a 10.75% drop off in accuracy when ResNet18 is evaluated on the validation set. In contrast, the same change only forces a 7.65% drop off for ECN.

Despite an ablation study on Efficient-CapsNet being absent in the literature, other investigations can explain our result. Our previous experiment concluded Efficient-CapsNet outperforms CNNs in all settings, with the difference most pronounced when data is limited. The literature (Mobiny & Van Nguyen, 2018; Jiménez-Sánchez et al., 2018) concluded the same result, although their implementation of capsule networks was CapsNet. Hence, in-line with previous results, we conclude Efficient-CapsNet alleviates some (but not all) of the need for data augmentation in a small-data setting. However, ECN does not alleviate the need at all when training data is large.



(i) 100% training set



(ii) 1% training set

Figure 3. The result of ResNet18 and Efficient-CapsNet when trained with different augmentation strategies in a full data and limited data environment.

5.3. Efficient-CapsNet and distributional generalization

To further support our experiments and offer greater insight into the generalization of Efficient-CapsNet, we use the concept of "distributional generalization" (Nakkiran & Bansal, 2020). The authors state two identical classifiers, trained under the same settings on disjoint training sets, should produce outputs belonging to the same distribution. Using this, they propose methods for assessing generalization that emphasize the distribution of the output of a set of classifiers rather than aggregated values such as average classification accuracy.

Since we used five training runs for each network and our small data regime was manufactured through sampling without replacement, we obtained five output distributions for each of our experiments. We use Nakkiran & Bansal (2020)'s agreement property for our study.

Definition (Agreement Property): *For classifiers f_1, f_2 trained on disjoint train sets S_1, S_2 sampled from some distribution D, the test accuracy of f_1 is close to its agreement probability with f_2. That is,*

$$P\big(f_1(\boldsymbol{x}) = y\big) \approx P\big(f_1(\boldsymbol{x}) = f_2(\boldsymbol{x})\big).$$

By considering the definition over an entire set with N samples $\boldsymbol{x}_n$, we define the *agreement probability* between two trained networks f_1, f_2 to be

$$A_{f_1,f_2} = \frac{1}{N}\sum_{n}^{N} \mathbb{1}_{f_1(\boldsymbol{x}_n)=f_2(\boldsymbol{x}_n)}. \tag{7}$$

Fig 4(i) shows the agreement probability A_{f_1,f_2} between each pair of classifiers f_1, f_2 and their average accuracy. We consider on our baseline and Efficient-CapsNet in order to focus solely on our first research question.

When full augmentation is applied during training (see Fig. 4(i)), all points from the 10% data regime (600 samples per class) are clustered for the baseline and ECN. This suggests both possess reasonable distributional generalization. In contrast, the models display different characteristics in the 1% data regime (60 samples per class). Data points associated with the baseline are distributed whilst those associated with ECN remain tightly clustered. This is significant since even in an extremely low data scenario, where accuracy is lower than

40%, Efficient-CapsNet continues to maintain distributional generalization.

However, we obtain different results in the absence of augmentation; see Fig. 4(ii). Unlike the full augmentation setting, baseline classifiers trained in the 10% regime lack distributional generalization, evidenced by the wide spread of points. This provides evidence that ECN alleviates the need for augmentation in order to generalize in the small-data setting.

5.4. Discussion

It is explained in Sec 4 that, given our input size, CapsNet produces $N_p = 2048$ primary capsules and Efficient-CapsNet produces $N_p = 16$ primary capsules. A graphical demonstration of their formation is provided in 5 and 6. It is remarkable Efficient-CapsNet outperforms CapsNet given the reduction in parameters by a factor of 128 in PrimaryCaps (and 47 across the network). Scaled-dot product attention has achieved excellent results in other fields (Vaswani et al., 2017; Devlin et al., 2019; Gong et al., 2021) and it is no different here. We suggest employing self-attention in the routing mechanism enables extremely informed decisions regarding the contribution of low-level capsules to high-level capsules. As a result, learning is significantly boosted despite the heavy reduction in parameters, even in a small-data regime.

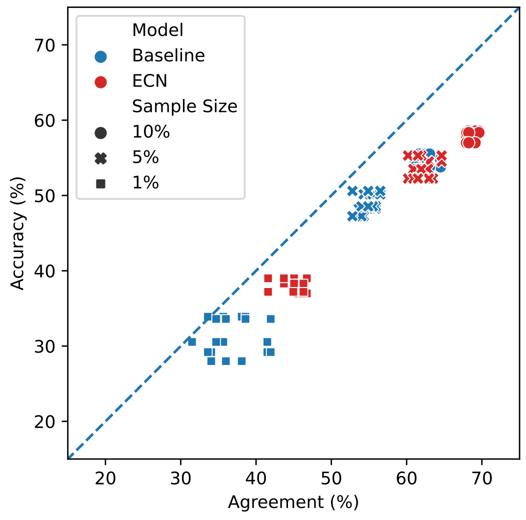
Despite the results suggesting Efficient-CapsNet promises a favourable alternative to CNNs in the low-parameter-small-data setting, we must consider our results in context. Achievement of 48.12% accuracy on the validation set when trained on 60 samples is far from the quality necessary to roll out the framework in developing countries' healthcare systems. Especially considering results here were obtained on CIFAR10 rather than more challenging medical datasets.

As outlined in Sec 1, to fill the research gap an assessment of Efficient-CapsNet's scalability must be performed. The reduced number of parameters in each layer will allow layers of attention routed capsules to be stacked (Mazzia et al., 2021), much like stacking in transformers (Vaswani et al., 2017). The highly parallelizable nature of self-attention routing means an increase in depth will not come at the expense of computational cost. To perform this investigation, we would treat depth as a hyperparameter and find optimal values for all other hyperparameters using initial convergence over 100 epochs, similar to here.

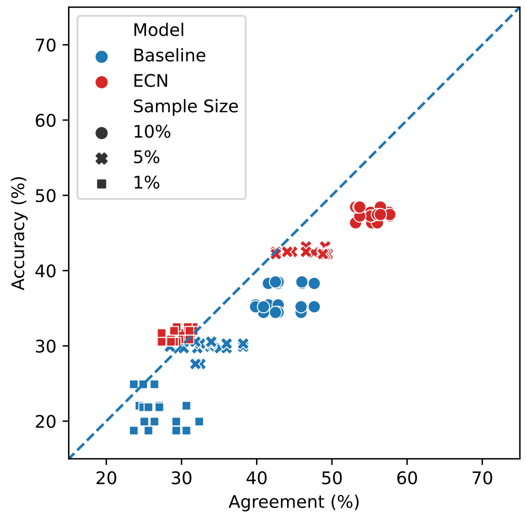
6. Conclusions

We found Efficient-CapsNet to outperform a low-parameter CNN, CapsNet, and ResNet18 in terms of classification accuracy in all data regimes. Most importantly, the gap between Efficient-CapsNet is most pronounced when the most extreme limitation on dataset size is presented and augmentation is absent. Efficient-CapsNet offers impressive distributional generalization in such a setting, even when classification accuracy is poor. Given large datasets and suitable augmentation strategies are hard to obtain for medical imaging datasets, there is very strong evidence to suggest Efficient-CapsNet should be pursued in the medical imaging community in developing countries.

Augmentation is found to benefit Efficient-CapsNet, and we recommend it is used in medical imaging tasks where possible.



(i) Full augmentation



(ii) No augmentation

Figure 4. Agreement probability, as defined in (7), versus average classification accuracy obtained by pairwise comparison of identically trained models on disjoint training sets.

Improvements in performance vary slightly with sample size, but this result is not considered significant.

As hinted in the introduction and outlined in Sec. 5.4, we believe further investigations should focus on scaling capsule networks. We used CIFAR10 to assess Efficient-CapsNet's ability to generalize when typical healthcare data challenges are imposed. Having obtained impressive results, we now suggest Efficient-CapsNet is scaled and applied to specific medical datasets. Rajasegaran et al. (2019) tried to scale the original CapsNet, but we believe self-attention routing should instead be the fundamental building block. The low number of parameters demanded and relatively relaxed training regime (in terms of computation) offers hope that the network can be scaled where computing resource is limited.

References

Abadi, Martín, Agarwal, Ashish, Barham, Paul, Brevdo, Eugene, Chen, Zhifeng, Citro, Craig, Corrado, Greg S., Davis, Andy, Dean, Jeffrey, Devin, Matthieu, Ghemawat, Sanjay, Goodfellow, Ian, Harp, Andrew, Irving, Geoffrey, Isard, Michael, Jia, Yangqing, Jozefowicz, Rafal, Kaiser, Lukasz, Kudlur, Manjunath, Levenberg, Josh, Mané, Dandelion, Monga, Rajat, Moore, Sherry, Murray, Derek, Olah, Chris, Schuster, Mike, Shlens, Jonathon, Steiner, Benoit, Sutskever, Ilya, Talwar, Kunal, Tucker, Paul, Vanhoucke, Vincent, Vasudevan, Vijay, Viégas, Fernanda, Vinyals, Oriol, Warden, Pete, Wattenberg, Martin, Wicke, Martin, Yu, Yuan, and Zheng, Xiaoqiang. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL https://www.tensorflow.org/. Software available from tensorflow.org.

Assiri, Yahia. Stochastic optimization of plain convolutional neural networks with simple methods, 2020.

Byerly, Adam, Kalganova, Tatiana, and Dear, Ian. No routing needed between capsules, 2021.

Chen, Liang-Chieh, Papandreou, George, Schroff, Florian, and Adam, Hartwig. Rethinking atrous convolution for semantic image segmentation. *arXiv preprint arXiv:1706.05587*, 2017.

Chollet, François. Xception: Deep learning with depthwise separable convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1251–1258, 2017.

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

Devlin, Jacob, Chang, Ming-Wei, Lee, Kenton, and Toutanova, Kristina. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1423. URL https://aclanthology.org/N19-1423.

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

Glorot, Xavier and Bengio, Yoshua. Understanding the difficulty of training deep feedforward neural networks. In Teh, Yee Whye and Titterington, Mike (eds.), *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, volume 9 of *Proceedings of Machine Learning Research*, pp. 249–256, Chia Laguna Resort, Sardinia, Italy, 13–15 May 2010. PMLR. URL https://proceedings.mlr.press/v9/glorot10a.html.

Gong, Yuan, Chung, Yu-An, and Glass, James. Ast: Audio spectrogram transformer, 2021.

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

Hinton, Geoffrey E, Krizhevsky, Alex, and Wang, Sida D. Transforming auto-encoders. In *Artificial Neural Networks and Machine Learning–ICANN 2011: 21st International Conference on Artificial Neural Networks, Espoo, Finland, June 14-17, 2011, Proceedings, Part I 21*, pp. 44–51. Springer, 2011.

Hinton, Geoffrey E, Sabour, Sara, and Frosst, Nicholas. Matrix capsules with em routing. In *International conference on learning representations*, 2018.

Jiménez-Sánchez, Amelia, Albarqouni, Shadi, and Mateus, Diana. Capsule networks against medical imaging data challenges. In *Intravascular Imaging and Computer Assisted Stenting and Large-Scale Annotation of Biomedical Data and Expert Label Synthesis: 7th Joint International Workshop, CVII-STENT 2018 and Third International Workshop, LABELS 2018, Held in Conjunction with MICCAI 2018, Granada, Spain, September 16, 2018, Proceedings 3*, pp. 150–160. Springer, 2018.

Jnawali, Kamal, Arbabshirani, Mohammad R, Rao, Navalgund, and Patel, Alpen A. Deep 3d convolution neural network for ct brain hemorrhage classification. In *Medical Imaging 2018: Computer-Aided Diagnosis*, volume 10575, pp. 307–313. SPIE, 2018.

Kauppi, Tomi, Kalesnykiene, Valentina, Kamarainen, Joni-Kristian, Lensu, Lasse, Sorri, Iiris, Raninen, Asta, Voutilainen, Raija, Uusitalo, Hannu, Kälviäinen, Heikki, and Pietilä, Juhani. The diaretdb1 diabetic retinopathy database and evaluation protocol. In *BMVC*, volume 1, pp. 10, 2007.

Kolesnikov, Alexander, Beyer, Lucas, Zhai, Xiaohua, Puigcerver, Joan, Yung, Jessica, Gelly, Sylvain, and Houlsby, Neil. Big transfer (bit): General visual representation learning. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part V 16*, pp. 491–507. Springer, 2020.

Koresh, H. James Deva, Chacko, Shanty, and Periyanayagi, M. A modified capsule network algorithm for oct corneal image segmentation. *Pattern Recognition Letters*, 143:104–112, 2021. ISSN 0167-8655. doi: https://doi.org/10.1016/j.patrec.2021.01.005. URL https://www.sciencedirect.com/science/article/pii/S0167865521000155.

Krizhevsky, Alex, Hinton, Geoffrey, et al. Learning multiple layers of features from tiny images. 2009.

Krizhevsky, Alex, Sutskever, Ilya, and Hinton, Geoffrey E. Imagenet classification with deep convolutional neural networks. In Pereira, F., Burges, C.J., Bottou, L., and Weinberger, K.Q. (eds.), *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012. URL https://proceedings.neurips.cc/paper_files/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf.

LeCun, Yann, Bottou, Léon, Bengio, Yoshua, and Haffner, Patrick. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Li, Qing, Cai, Weidong, Wang, Xiaogang, Zhou, Yun, Feng, David Dagan, and Chen, Mei. Medical image classification with convolutional neural network. In *2014 13th international conference on control automation robotics & vision (ICARCV)*, pp. 844–848. IEEE, 2014.

Lin, Ancheng, Li, Jun, and Ma, Zhenyuan. On learning and learned data representation by capsule networks. *arXiv preprint arXiv:1810.04041*, 2018.

Loshchilov, Ilya and Hutter, Frank. SGDR: Stochastic gradient descent with warm restarts. In *International Conference on Learning Representations*, 2017. URL https://openreview.net/forum?id=Skq89Scxx.

Loshchilov, Ilya and Hutter, Frank. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=Bkg6RiCqY7.

Mazzia, Vittorio, Salvetti, Francesco, and Chiaberge, Marcello. Efficient-capsnet: Capsule network with self-attention routing. *Scientific reports*, 11(1):14634, 2021.

Mobiny, Aryan and Van Nguyen, Hien. Fast capsnet for lung cancer screening. In *Medical Image Computing and Computer Assisted Intervention–MICCAI 2018: 21st International Conference, Granada, Spain, September 16-20, 2018, Proceedings, Part II 11*, pp. 741–749. Springer, 2018.

Nakkiran, Preetum and Bansal, Yamini. Distributional generalization: A new kind of generalization. *arXiv preprint arXiv:2009.08092*, 2020.

Pan, Chenbin and Velipasalar, Senem. Pt-capsnet: A novel prediction-tuning capsule network suitable for deeper architectures. In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 11976–11985, 2021. doi: 10.1109/ICCV48922.2021.01178.

Patrick, Mensah Kwabena, Adekoya, Adebayo Felix, Mighty, Ayidzoe Abra, and Edward, Baagyire Y. Capsule networks–a survey. *Journal of King Saud University-computer and information sciences*, 34(1):1295–1310, 2022.

Rajasegaran, Jathushan, Jayasundara, Vinoj, Jayasekara, Sandaru, Jayasekara, Hirunima, Seneviratne, Suranga, and Rodrigo, Ranga. Deepcaps: Going deeper with capsule networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 10725–10733, 2019.

Ramasinghe, Sameera, Athuraliya, CD, and Khan, Salman H. A context-aware capsule network for multi-label classification. In *Proceedings of the European Conference on Computer Vision (ECCV) Workshops*, pp. 0–0, 2018.

Recht, Benjamin, Roelofs, Rebecca, Schmidt, Ludwig, and Shankar, Vaishaal. Do cifar-10 classifiers generalize to cifar-10?, 2018.

Recht, Benjamin, Roelofs, Rebecca, Schmidt, Ludwig, and Shankar, Vaishaal. Do ImageNet classifiers generalize to ImageNet? In Chaudhuri, Kamalika and Salakhutdinov, Ruslan (eds.), *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 5389–5400. PMLR, 09–15 Jun 2019. URL https://proceedings.mlr.press/v97/recht19a.html.

Ren, Hao, Su, Jianlin, and Lu, Hong. Evaluating generalization ability of convolutional neural networks and capsule networks for image classification via top-2 classification. *arXiv preprint arXiv:1901.10112*, 2019.

Ren, Shaoqing, He, Kaiming, Girshick, Ross, and Sun, Jian. Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems*, 28, 2015.

Ronneberger, Olaf, Fischer, Philipp, and Brox, Thomas. U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III 18*, pp. 234–241. Springer, 2015.

Sabour, Sara, Frosst, Nicholas, and Hinton, Geoffrey E. Dynamic routing between capsules. *Advances in neural information processing systems*, 30, 2017.

Simonyan, Karen and Zisserman, Andrew. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

Spanhol, Fabio Alexandre, Oliveira, Luiz S, Petitjean, Caroline, and Heutte, Laurent. Breast cancer histopathological image classification using convolutional neural networks. In *2016 international joint conference on neural networks (IJCNN)*, pp. 2560–2567. IEEE, 2016.

Srivastava, Nitish, Hinton, Geoffrey, Krizhevsky, Alex, Sutskever, Ilya, and Salakhutdinov, Ruslan. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958, 2014. URL http://jmlr.org/papers/v15/srivastava14a.html.

Tan, Mingxing and Le, Quoc V. Efficientnetv2: Smaller models and faster training, 2021.

Vaswani, Ashish, Shazeer, Noam, Parmar, Niki, Uszkoreit, Jakob, Jones, Llion, Gomez, Aidan N, Kaiser, Łukasz, and Polosukhin, Illia. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Veta, Mitko, Heng, Yujing J., Stathonikos, Nikolas, Bejnordi, Babak Ehteshami, Beca, Francisco, Wollmann, Thomas, Rohr, Karl, Shah, Manan A., Wang, Dayong, Rousson, Mikael, Hedlund, Martin, Tellez, David, Ciompi, Francesco, Zerhouni, Erwan, Lanyi, David, Viana, Matheus, Kovalev, Vassili, Liauchuk, Vitali, Phoulady, Hady Ahmady, Qaiser, Talha, Graham, Simon, Rajpoot, Nasir, Sjöblom, Erik, Molin, Jesper, Paeng, Kyunghyun, Hwang, Sangheum, Park, Sunggyun, Jia, Zhipeng, Chang, Eric I-Chao, Xu, Yan, Beck, Andrew H., van Diest, Paul J., and Pluim, Josien P.W. Predicting breast tumor proliferation from whole-slide images: The tupac16 challenge. *Medical Image Analysis*, 54:111–121, 2019. ISSN 1361-8415. doi: https://doi.org/10.1016/j.media.2019.02.012. URL https://www.sciencedirect.com/science/article/pii/S1361841518305231.

Vinyals, Oriol, Blundell, Charles, Lillicrap, Timothy, Wierstra, Daan, et al. Matching networks for one shot learning. *Advances in neural information processing systems*, 29, 2016.

Wan, Li, Zeiler, Matthew, Zhang, Sixin, Le Cun, Yann, and Fergus, Rob. Regularization of neural networks using dropconnect. In *International conference on machine learning*, pp. 1058–1066. PMLR, 2013.

Wood, Luke, Tan, Zhenyu, Ian, Stenbit, Zhu, Scott, Chollet, François, et al. Kerascv. https://github.com/keras-team/keras-cv, 2022.

Wu, Haiping, Xiao, Bin, Codella, Noel, Liu, Mengchen, Dai, Xiyang, Yuan, Lu, and Zhang, Lei. Cvt: Introducing convolutions to vision transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 22–31, 2021.

Xi, Edgar, Bing, Selina, and Jin, Yang. Capsule network performance on complex data. *arXiv preprint arXiv:1712.03480*, 2017.

Zhao, Zhen, Kleinhans, Ashley, Sandhu, Gursharan, Patel, Ishan, and Unnikrishnan, K. P. Capsule networks with max-min normalization, 2019.

A. Results of experiments

Sample size (%)	Set	Baseline	ResNet18	CapsNet	ECN	Gap
100	Val	85.28	85.28	79.51*	**86.11**	0.83
	Test	74.27	73.21	65.45	**75.34**	1.07
10	Val	68.24	65.48	64.03	**71.43**	3.19
	Test	54.69	51.16	50.38	**57.91**	3.22
5	Val	62.19	53.13	58.40	**66.12**	3.93
	Test	48.96	40.95	45.56	**53.26**	4.30
1	Val	39.73	35.49	41.60	**48.12**	8.39
	Test	31.05	27.88	32.19	**37.90**	6.85

Table 2. Classification accuracy (%) for models trained on CIFAR10 validation set (Krizhevsky et al., 2009) and CIFAR10 test set (Recht et al., 2018; 2019). Gap is the difference between Efficient-CapsNet and the baseline. * indicates we were unable to reproduce a result stated in the original paper. The full training set is sampled without replacement to produce disjoint training sets when sample size < 100%.

Sample size (%)	Augmentation	ResNet18	ECN	Gap
100	None	74.91	**75.93**	1.02
100	Random Crop	80.02	**83.37**	3.35
	Color Jitter	75.80	**76.16**	0.36
	Horizontal Flip	**78.74**	78.57	-0.17
	Cutout	75.24	**79.32**	4.08
100	All	85.55	**86.34**	0.79
1	None	22.75	**40.43**	17.68
1	Random Crop	30.00	**43.75**	13.75
	Color Jitter	22.13	**39.71**	17.58
	Horizontal Flip	25.01	**42.24**	17.23
	Cutout	24.76	**38.40**	13.64
1	All	33.53	**48.02**	14.49

Table 3. Classification accuracy (%) for models trained on CIFAR10 validation set (Krizhevsky et al., 2009) with varying augmentation strategies. Gap is the difference between Efficient-CapsNet and ResNet18. The full training set is sampled without replacement to produce disjoint training sets when sample size < 100%.

B. Model architectures

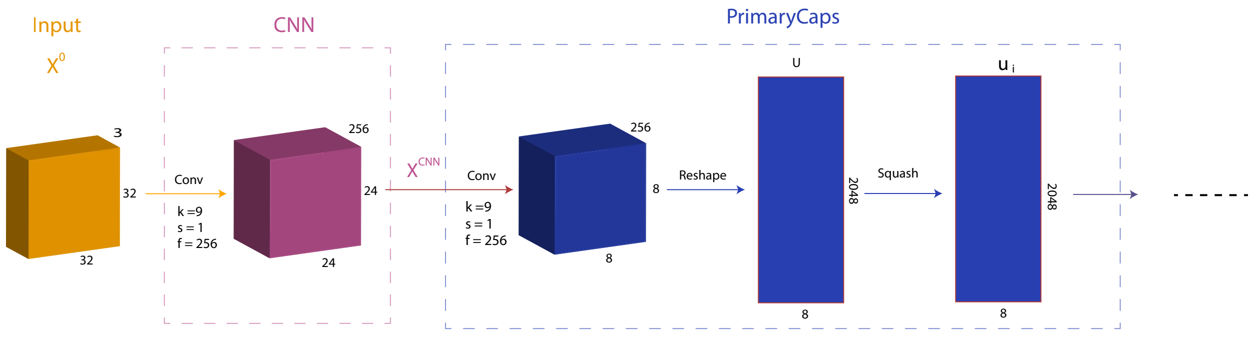


Figure 5. The model architecture for the CNN and PrimaryCaps components in CapsNet shows the formation of 2048 primary capsules with dimension 8 (given the 32×32 image size in CIFAR10).

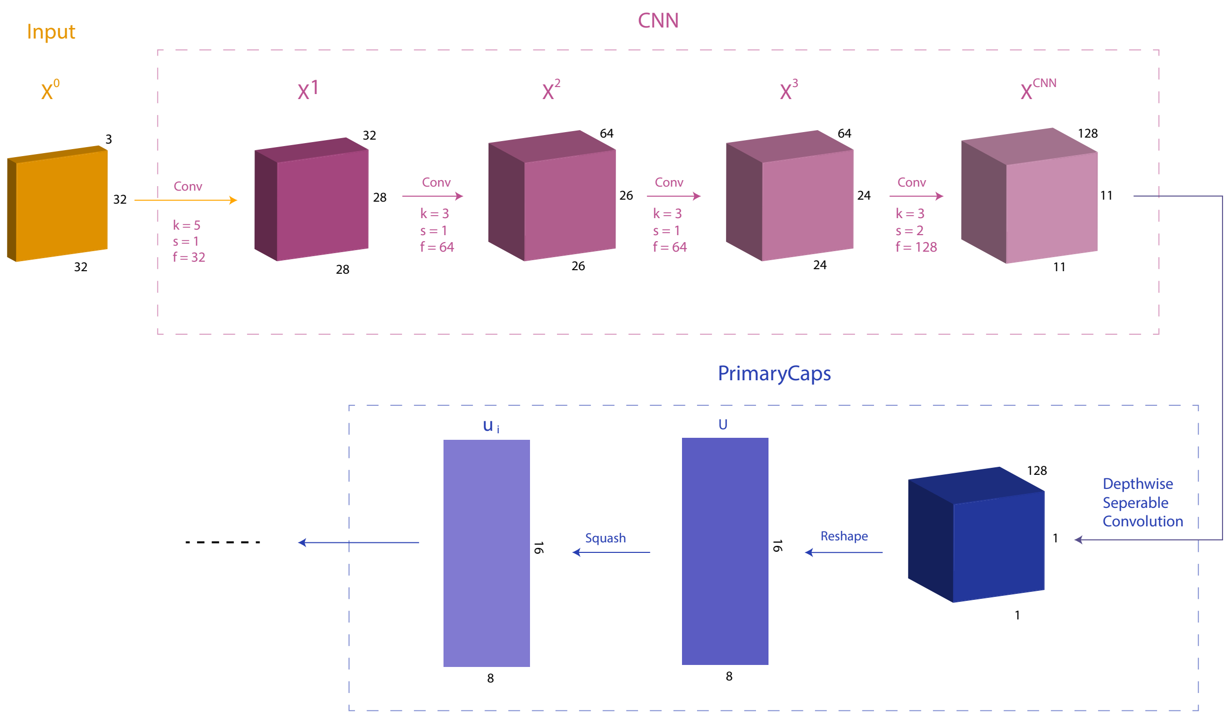


Figure 6. The model architecture for the CNN and PrimaryCaps components in Efficient-CapsNet shows the formation of 16 primary capsules with dimension 8 (given the 32×32 image size in CIFAR10). The depthwise separable convolution operation is explained in 4.3.1.

C. Other figures

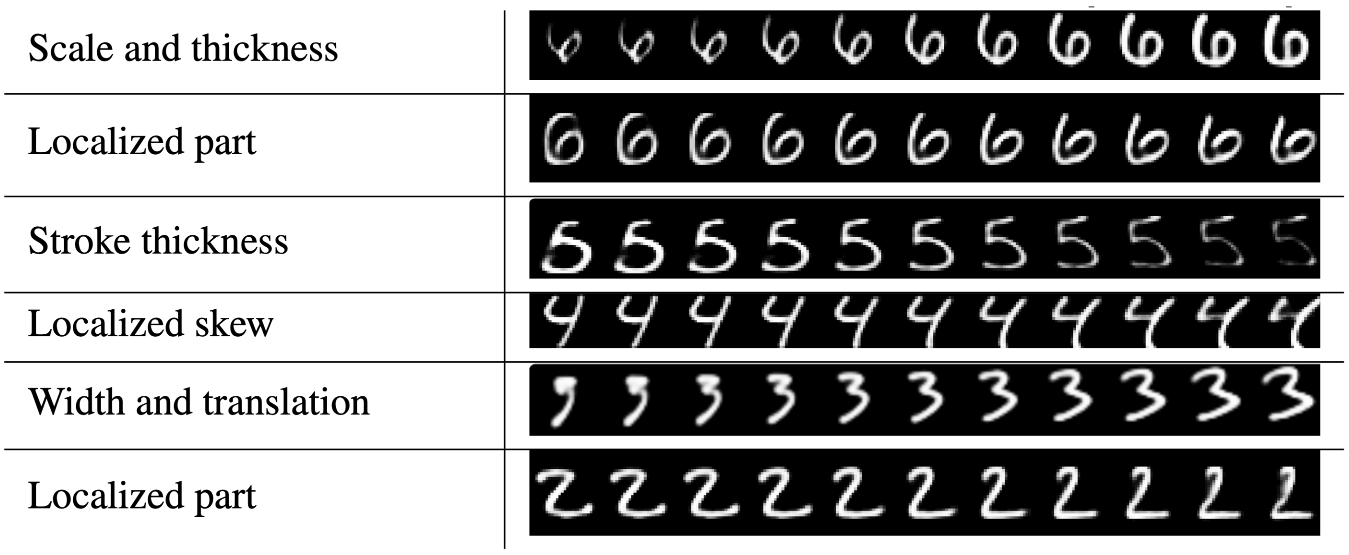


Figure 7. Dimension perturbations; Figure 4 in Sabour et al. (2017). The visualization shows the reconstruction after passing input samples through the capsule-based encoder and corresponding decoder. Tweaking each dimension of the encoded embedding by intervals of 0.05 in the range $[-0.25, 0.25]$ shows the correspondence between a dimension of the output vector and the appearance of the image.