THE UNIVERSITY OF MANCHESTER

BSc DISSERTATION

Correlation and attention in artificial intelligence

Rutwik Mudholkar
10327919

December 2020

Abstract

Attention is an increasingly popular mechanism used in a wide range of neural architectures. It allows us to visualise their inner workings, a notoriously difficult task for an effective black box, and subsequently make better predictions for classification problems. Here we lay out the basic structure of a Convolution Neural Network optimised for Computer Vision image processing, and how an Attention Mechanism is implemented therein. We discuss different Attention Mechanisms, comparing their merits and use cases. We also discuss possible statistical theory underpinning the relation of Attention Mechanisms to measures of similarity and correlation to gain a deeper mathematical understanding of Attention.

Contents

1 Introduction **3**

2 Artificial Neural Networks **3**

2.1 Neurons 3

2.2 Feedforward Neural Networks 4

3 Convolutional Neural Networks **6**

3.1 High-Level Feature Extraction 6

3.2 Training and Classification 7

4 Attention **7**

4.1 Initution 7

4.2 Implementation 8

5 Compatibility and Correlation **9**

5.1 Multiplicative Compatibility 9

5.2 Additive Compatibility 12

5.3 Significance 13

6 Conclusion **13**

1 Introduction

Deep Learning, as a branch of Machine Learning, is closely related to a class of theories of brain development proposed by cognitive neuroscientists in the early 1990s [1]. Many proposed the neural networks used in deep learning models were analogous to learning dynamics in the brain. For instance, neural networks employing a hierarchy of layered filters in which each layer considers information from a prior layer, and then passes its output to other layers, is similar to the function of the cortex [2]. The Convolutional Neural Network (CNN) in particular is a well-known deep learning architecture additionally inspired by the natural visual perception mechanism of living creatures. Computer vision based on CNNs has enabled people to accomplish what had been considered impossible in the past few centuries, such as face recognition, autonomous vehicles, self-service supermarkets, and intelligent medical treatment [3].

Neuroscientists have long studied Attention as an important cognitive process. It is described as the ability of organisms to 'select a subset of available information upon which to focus for enhanced processing and integration' [4]. The computational equivalent equips a neural network with the ability to focus on a subset of its inputs. The concept of 'Attention' has a long history in classical computer vision, where an Attention Mechanism selects relevant parts of the image for object recognition in cluttered scenes [5]. Paying attention to Attention is important for todays data heavy and computer intensive learning algorithms, as it provides essential cues for downstream Computer Vision models and frees up resources to focus on the most task-related parts of inputs [6]. In particular, understanding the mathematics of Attention Functions and how they measure the similarity, or correlation of relevant image parts may allow us to delve deeper into why they are so effective.

Implementation of Attention can be done by analysing intermediate layers in a neural network pipeline. This dissertation will review different approaches to measuring correlations of these layers, and how they map to the different Attention Functions in CNNs.

2 Artificial Neural Networks

2.1 Neurons

Artificial Neural Networks (ANNs) are comprised of many interconnected computational nodes (referred to as neurons), which work in a distributed fashion to collectively learn from the input to optimise its final output [7][8]. The basic unit of computation in a neural network is the neuron, often called a node or unit. It receives input from some other nodes, or from an external source and computes an output.

The output Y from the neuron is computed as

$$Y = f(w_1x_1 + w_2x_2 + ... + w_nx_n) \tag{1}$$

where x_i are the numerical inputs, w_i are the connection weights, and f is non-linear and is called the Activation Function. The purpose of the Activation Function is to introduce non-linearity into

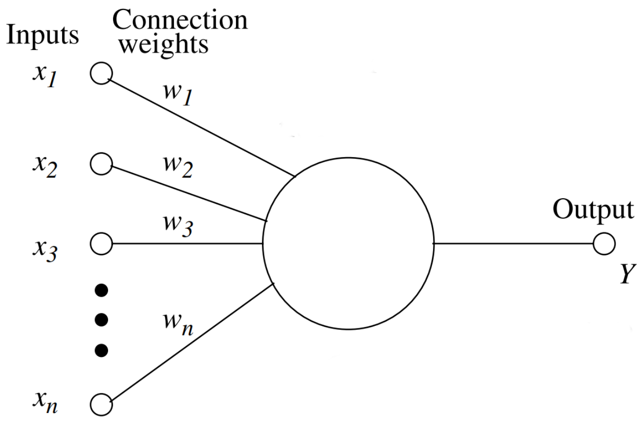

Figure 1: Diagrammatic representation of a neuron adapted from [9]. It receives numerical inputs $x_1, x_2, ...x_n$ with respective weights $w_1, w_2, ...w_n$ and outputs Y.

the output of a neuron. This is important since real world data is non linear by nature, so we want our model to learn those representations [10].

2.2 Feedforward Neural Networks

Feedforward Neural Networks (FFFNNs) were the first and simplest type of ANNs devised [11]. They contain multiple neurons (also called nodes) arranged in layers. Nodes from adjacent layers have connections between them. All these connections have weights associated with them; given an input vector, they are the variables that determine what the output vector is.

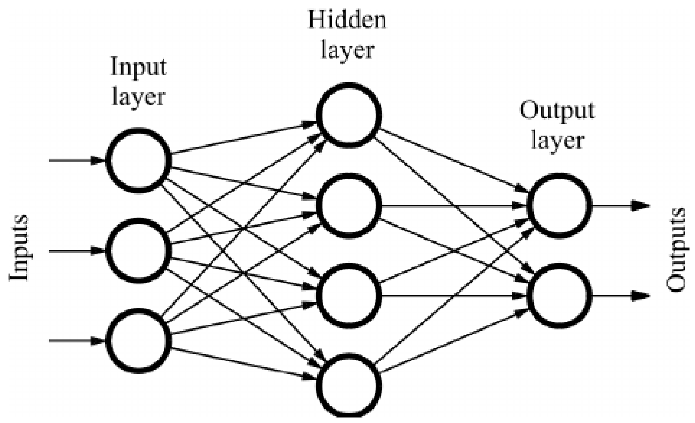

Figure 2: Example of FFNN reproduced from [12]. It takes in 3 inputs and returns 1 of 2 outputs. w_{ij}^h is the weight for a connection from the ith node in the hth layer, to the jth node in the next layer (weights not shown).

Figure 2 shows a simple example of a FFNN. The input nodes provide data to the network and are together referred to as the 'Input Layer'. There is no computation at this stage; the data is simply piped to the hidden nodes. A collection of hidden nodes forms a 'Hidden Layer'. While a FFNN will only have a single Input Layer and a single Output Layer, it can have zero or multiple Hidden Layers. The Hidden Layers perform computations and pass on their outputs to the next Hidden Layer. The final Hidden Layer transfers its outputs to the 'Output Layer', which output the net result of the network. The information always moves in the forward direction, and there are no loops or cycles [13]. A FFNN with one or more Hidden Layers is called a Multi Layer Perceptron (MLP).

Since we are interested in classification tasks using CNNs, we use an Activation Function in the Output Layer of the MLP such that the outputs are probabilities that sum to 1, such as a Softmax [14]. Each output node is associated with some label e.g. “dog”, “cat” etc. as classifications of an input e.g. an image of an animal. That way, given a vector of inputs $\boldsymbol{X} = (x_1, x_2, ..., x_i)$ that give a vector of outputs $\boldsymbol{Y} = (y_1, y_2, ..., y_j)$, the classification

$$C = label(k),\ y_k = max(\boldsymbol{Y}) \tag{2}$$

Since the initial weights are usually assigned randomly, The final output for an MLP has to be optimised through ‘learning’. The goal of learning is to assign correct weights for the connections. This can be done via ‘Backward Propagation of Errors’, defined as ‘repeatedly adjusts the weights of the connections in the network so as to minimize a measure of the difference between the actual output vector of the net and the desired output vector’ [15]. This difference is the loss L and is calculated via a loss function $L = f(\boldsymbol{Y}, \hat{\boldsymbol{Y}})$ where $\hat{\boldsymbol{Y}}$ is the desired output. Generally, $\hat{y_i} = \delta_{i,j}$ where j represents the desired label. In order to minimize L, it is propagated backward through the network to calculate the gradients

$$\frac{\partial L}{\partial \boldsymbol{W}} = \{\frac{\partial L}{\partial w^h_{ij}}\} \tag{3}$$

with respect to the weights. In a process analogous to numerically solving for the minimum of a curve, stochastic gradient descent is used to adjust the weights by

$$w^h_{ij} \to w^h_{ij} - \epsilon \frac{\partial L}{\partial w^h_{ij}}. \tag{4}$$

The choice of L is contingent on the model type. For classification, Cross-entropy loss is preferred [16]. Backpropagation is usually used in conjunction with Whitening transformations to reduce Internal Covariate Shift [17], the change in the distribution of network activations due to the change in network parameters during training. This can be achieved by standardising and scaling input data, such that the mean $\boldsymbol{\mu}_i$ of each input $\boldsymbol{X}_i$ equals 0, and that the covariances between each all the inputs are about the same [18]. Whitening transformations can be affine such as the Mahalanobis Transformation:

$$\boldsymbol{z} = \boldsymbol{\Sigma}^{-1/2}(\boldsymbol{x} - \boldsymbol{\mu}) \tag{5}$$

where $\boldsymbol{\Sigma}, \boldsymbol{\mu}$ is the covariance matrix and mean vector for data vector $\boldsymbol{x}$. The univariate case for a scalar x reduces to

$$z = \frac{x - \mu}{\sigma}, \tag{6}$$

which is the familiar form of a standardised normally distributed variable. These steps are taken in a traditional neural network to speed up the convergence of $L \to 0$, but for the case of an Attention based CNN, it also helps with finding an expression for Compatibility (Section 4). A MLP is typically the last step of a CNN pipeline.

3 Convolutional Neural Networks

CNNs are a specialized type of neural network model designed for working with two-dimensional image data [19]. LeNet was one of the very first and simplest CNNs [20] and is sufficient to explain the implementation of Attention.

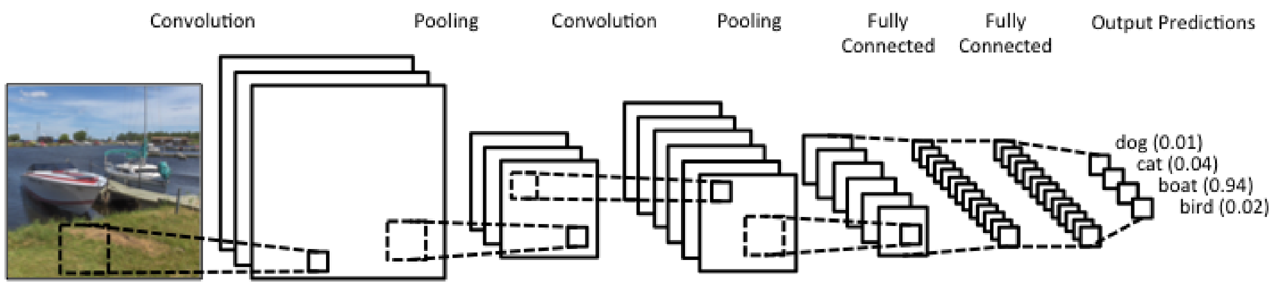

Figure 3: Visualisation of the LeNET architecture reproduced from [21]. The ReLU steps are not shown.

3.1 High-Level Feature Extraction

The three main levels for this stage are Convolutions, introducing Non Linearity, and Pooling. A Convolution (Fig. 3) is a linear operation involving an input tensor $\boldsymbol{I}$ and a Kernel tensor $\boldsymbol{K}$ to produce an 2D matrix $\boldsymbol{S}$ known as the 'Feature Map', which can represented as

$$\boldsymbol{S}_{i,j} = \sum_n \sum_x \sum_y I_{i+x-1,j+y-1,n} \cdot \boldsymbol{K}_{x,y,n}. \tag{7}$$

where x, y are the dimensions of $\boldsymbol{K}$, n is the depth of the tensor and the stride length is 1. In the case of $\boldsymbol{I}$ being an original colour image, it is split into 3 2D matrices using its colour channels (R,G,B), where each element is the channel pixel value. The matrices are concatenated to form a tensor of depth $n = 3$. To retrieve each element of $\boldsymbol{S}$, $\boldsymbol{K}$ 'strides' across $\boldsymbol{I}$ and performs the operation in Equation 4. $\boldsymbol{K}$ must be smaller than $\boldsymbol{I}$, and the specific size of $\boldsymbol{K}$ is decided prior based on certain factors [22]. The Convolution operation reduces matrix size since $\boldsymbol{K}$ cannot breach the boundary of $\boldsymbol{I}$, so if we want to ensure $\boldsymbol{S}$ and $\boldsymbol{I}$ have equal dimensionality, $\boldsymbol{I}$ is 'padded' with additional row and columns to increase its size [23]. Multiple Kernels can be used in one convolution layer to produces multiple Feature Maps, which are concatenated to form the new tensor.

A nonlinear element wise operation is applied to the resulting Feature Map to introduce nonlinearlity into our CNN (see Section 2.1). One example is a Rectified Linear Unit (ReLU), given by

$$\boldsymbol{S}_{i,j} \to max(\boldsymbol{S}_{i,j}, 0) \tag{8}$$

Other non linear functions such as tanh or sigmoid can also be used [24]. Spatial Pooling is then used to reduce the number of parameters and amount of computation in the network, hence controlling over-fitting while still retaining key features. In case of Max Pooling [25], we define a spatial neighborhood (for example, a 2×2 window) and take the largest element from the rectified Feature Map within that window. The window $\boldsymbol{W}$ strides across $\boldsymbol{S}$ in a similar way to the Convolutional operation, however the stride length is now equal to the length of $\boldsymbol{W}$. Spatial Pooling can also be a Sum or Average. The Convolution, Nonlinear and Pooling steps can be successively performed multiple times, transforming $\boldsymbol{I} \rightarrow \{\boldsymbol{S}^1\} \rightarrow \{\boldsymbol{S}^2\} \rightarrow ... \rightarrow \{\boldsymbol{S}^L\}$ where $\{\boldsymbol{S}^l\}$ represents the set of Feature Maps at the lth layer of L total layers along the CNN pipeline.

3.2 Training and Classification

The training process for a CNN is similar to the Backpropagation of Errors and gradient descent model for a standard MLP (Section 2.2). The parameters learned are the connection weights for the Fully Connected MLP (FC) layers, and the values of the Kernel Filters. In practice, the set of Kernels that operate directly on the raw pixel values learn to extract low-level features, and sets operating on Feature Maps in subsequent layers learn to extract features that are combinations of lower-level features, such as lines and shapes. An example of a low-level Kernel Filter is that for for edge detection, which can take the form of

$$\boldsymbol{K} = \begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix} \tag{9}$$

for the case $\boldsymbol{K} \in \mathbb{R}^{3\times3}$. The resulting Feature Map would represent an image with sharper boundaries between objects and different colours, useful for extracting simple lines. This process continues until very deep layers are extracting faces, animals, and other complex objects [26][27]. The final set of Feature Maps $\{S_n\}$ is turned into a 1D vector which is piped through a series of FC layers for classification. Figure 3 shows the original image being correctly classified as a 'boat' with $P(boat) = 1$

4 Attention

4.1 Inituition

Extracting and choosing suitable features is crucial for image processing [28]. However, this is challenging for images with complex backgrounds [29]. One reason is due to the black box nature of CNNs; the visual reasoning they implement through their pipelines in solving problems remains largely inscrutable. A way to visualise and interpret the inner workings of CNNs, and thus extract useful data at intermediate layers throughout the process, is by the implementation of Attention, specifically finding the Attention Map. The Attention Map is a scalar matrix representing the relative importance of layer activations at different 2D spatial locations with respect to the target task [30]. The Maps are used to exploit the effective spatial support of Feature Maps used by CNNs, based on the idea that there is benefit to identifying salient image regions and amplifying their influence, while likewise suppressing the irrelevant and potentially confusing information in other regions. Attention can be incorporated into any existing CNN architecture, including LeNet.

One method to implement Attention-aware classification is by looking at the Compatibility between local feature vectors extracted at intermediate layers in the CNN, and the final global feature vector normally fed into the FC classification layers. Since this global feature vector represents salient higher-order visual concepts in different dimensions, this is achieved by encouraging our local feature vectors to learn similar mappings [31]. Different kinds of class details are more easily accessible at different scales, so in order to facilitate the learning of diverse and complementary Attention-weighted features, layers are chosen based on different spatial resolutions. Rather than classify using solely this final feature vector like in a traditional CNN, we can restrict the classifier to instead use the intermediate local feature vectors, chosen and weighted by their Compatibility scores. The hyper-parameter here is the choice of Compatibility function, and is discussed further down.

4.2 Implementation

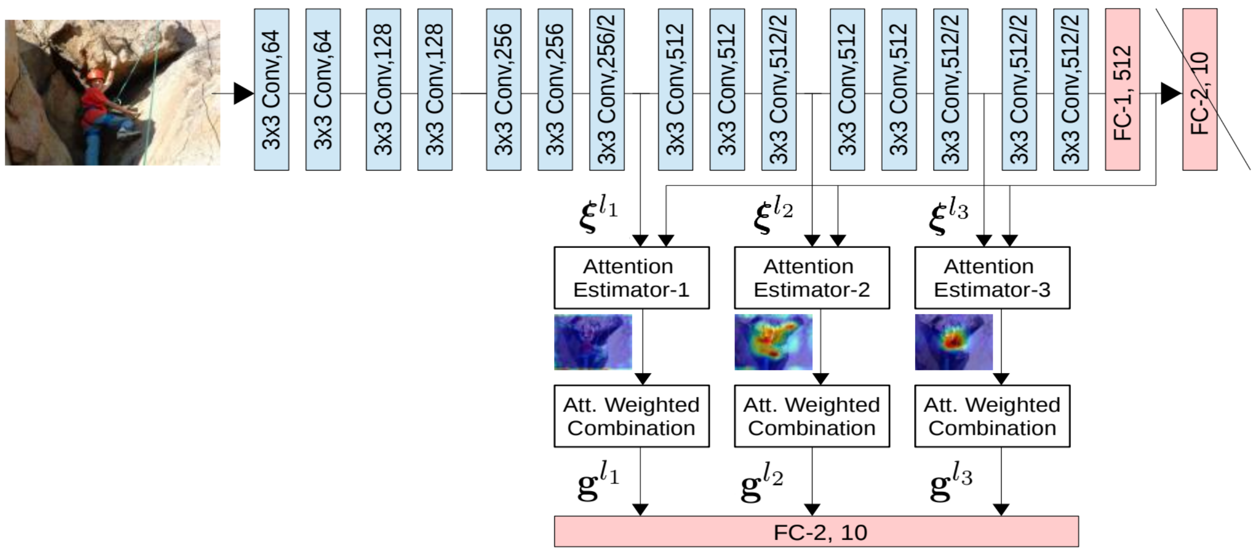

Figure 4: Visualisation of Attention incorporated into a CNN architecture adapted from [31]. Attention is introduced at 3 distinct layers, with the lowest level Attention Maps focusing on the surroundings (the rocks), intermediate level maps on object parts (harness and climbing equipment) and the highest level maps on the central object.

Consider the set of 2D Feature Maps $\{\boldsymbol{S}^l\}$ at layer l as a 3D tensor $\boldsymbol{\xi}^l$ where F_l is the depth, equal to the number of Feature Maps. $\boldsymbol{\xi}^l \equiv \{\boldsymbol{s}^l_k\}^n_{k=1}$, where each $\boldsymbol{s}^l_k$ represents the pixel-wise feature vector of length F_l at the kth spatial location. Let $\boldsymbol{g} \in \mathbb{R}^{F_g}$ be the global feature vector extracted just before the final classification layer of a CNN. It has the entire input image as support and is output by the network's series of convolutional and nonlinear layers. In this case $\boldsymbol{g}$ encodes global, discriminative, and relevant information about the objects of interest [32]. We can now consider individual Attention Gates (AGs) at distinct intermediate layers to compute their Compatibility with $\boldsymbol{g}$. In order to produce the set of Compatibility scores $\{c^l_k\}$ at layer l, we use some Compatibility function $C(\boldsymbol{\xi}^l, \boldsymbol{g})$. In the case where $\boldsymbol{s}^l_k$ and $\boldsymbol{g}$ have different lengths, a learnable weight matrix $\boldsymbol{W}_g \in \mathbb{R}^{F_l \times F_g}$ can be used to match their lengths.

For each $\boldsymbol{s}^l_k$, the AG then computes Attention Coefficients $A^l \equiv \{a^l_k\}^n_{k=1}$ where $a^l_k \in [0, 1]$, by normalising $\{c^l_k\}$ with a softmax function:

$$a^l_k = \frac{\exp(c^l_k)}{\sum_j \exp(c^l_j)}. \tag{10}$$

The output of an AG at layer l is given by a weighted sum $\boldsymbol{g}^l = \sum_k a^l_k \cdot \boldsymbol{s}^l_k$. The higher the c^l_k, the higher the a_k^l, meaning the corresponding $\boldsymbol{s}_k^l$ will be relatively weighted higher and have a stronger contribution to $\boldsymbol{g}^l$. The effect of an exponential rather than linear calculation greatly exaggerates small differences, ensuring $\boldsymbol{g}^l$ has minimal noise. Softmax also enforces $0 \leq a_k^l \leq 1 \; \forall \in \{1 \ldots n\}$ and $\sum_i a_i = 1$, so the combination of feature vectors is convex. This ensures that features at different spatial locations must effectively compete against one another for their share of the Attention Map. We can now constrain our final FC inputs to a new set $\{\boldsymbol{g}^l\}$ from all AGs, forcing our network to learn the most salient features that contribute to the class. Since the Attention Mechanism has access to the entire image, we call it 'Soft' Attention, since the learned alignment weights a_k^l are placed 'softly' over all patches in the source image. This makes the image smooth and differentiable, but computationally expensive when the source image is large. 'Hard' Attention [33] on the other hand only selects one patch of the image to attend to at a time, making it cheaper, but requiring more complicated techniques such as variance reduction or reinforcement learning to train.

5 Compatibility and Correlation

$C(\boldsymbol{\xi}^l, \boldsymbol{g})$ is one of the most important hyper-parameters that differs in literature based on various input image attributes. The two groups of Compatibility functions used in computer vision based CNNs are Multiplicative and Additive Compatibility (MC, AC respectively). However, how these exactly translate to mathematical strengths of relation and similar mappings is scarcely studied in literature. The method by which we vary the Attention Mechanism will be solely varying $C(\boldsymbol{\xi}^l, \boldsymbol{g})$, since it is the only step which compares the two vectors.

5.1 Multiplicative Compatibility

MC has the general form

$$c_k^l = \boldsymbol{s}_k^l \cdot \boldsymbol{g} \tag{11}$$

as used in [31]. We could interpret this as the unnormalized form of cosine similarity, which measures the degree of alignment of two vectors. We can see it resembles Pearsons correlation coefficient (PCC) [34], which uses the normalised form of the co-variance, which itself is a dot product between the deviations of two raw data sets from their mean. MC is a raw dot product, however. Consider 3 channel colour images with pixel values ranging between 0 and 255, with feature vectors $\boldsymbol{s}_k^l$ having $\boldsymbol{\mu}_i > \boldsymbol{0}$ and covariance $\boldsymbol{\Sigma}$. The more $\boldsymbol{\mu}_i > 0$ strays from 0, the less MC may represent correlation. This issue is noticeably amplified in the case $\boldsymbol{\mu}_i \neq \boldsymbol{\mu}_j, \delta_{ij} = 0$, where $\boldsymbol{s}_k^l$ with higher means will naturally give higher dot products, regardless of correlation. This issue can be resolved by the zero centering normalisation of input data done by whitening of the feature vectors at each layer during backpropagation (see Section 2.2). In this case $\boldsymbol{\mu} = \boldsymbol{0}$, reducing MC to PCC. We can therefore see that MC represents mapping similarity via correlation.

From Figure 5 we can see this relation. $\boldsymbol{\mu} = \boldsymbol{0}$, and $\boldsymbol{\Sigma}$ is randomly generated and multiplied by a scalar $\epsilon \sim 10^{-3}$ to approximate the whitening covariance similarity condition. Here, c increases proportionally to the perceived strength of positive correlation between the two vectors. Inserting the set of Compatibility scores into Equation 8 will yield exponentially increasingly larger attention Coefficients for plots 4 to 9, and yield effectively 0 for plots 1 to 3. The Attention Coefficient ratio between plots 8 and 9 is $e^{676.22}/e^{673.3} \sim 10^1$, and between 7 and 8 is $e^{673.3}/e^{463.8} \sim 10^{90}$, a huge

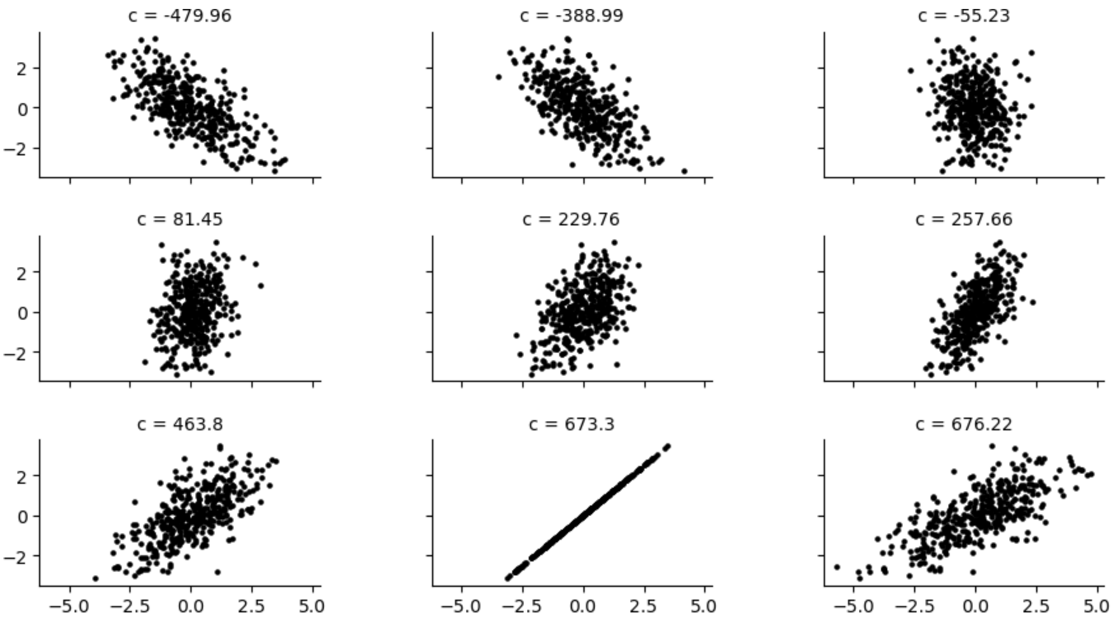

Figure 5: Plots 1 to 9 of vectors $\{\boldsymbol{v}\} \sim N(\mathbf{0}, \boldsymbol{\Sigma}), |\{\boldsymbol{v}\}| = 9$ against a vector $\boldsymbol{v}_k \in \{\boldsymbol{v}\}, k = 5$ with their Compatibility scores c calculated using Equation 11. Plot 8 is vector $\boldsymbol{v}_k$ against itself

magnitudinal difference that leaves only 8 and 9 contributing to $\boldsymbol{g}^l$. While still Soft Attention, it borders on Hard Attention, and will produce a sharp Attention Map. We see a discrepancy with plots 8 and 9, where a noticeably lower correlation generates a higher Compatibility score. This can be explained by edge conditions, wherein the range of pixel values deviates further from the mean in plot 9, likely due to the small fluctuations in $\boldsymbol{\Sigma}$. This can be resolved by restricting the range or further equalising the co-variances through back-propagation. It may also be minimized by only extracting Feature Maps at deep layers, where the numerous Convolutional filters beforehand have removed pixel noise and made them sparse. Sparse Feature Maps have many pixels close or equal to 0 ('deactivated') while the others have high activations, a natural consequence to extracting salient features. This would result in the data within $\boldsymbol{s}_k^l$ resembling a more discrete distribution. In this case, only mutually activated pixel vector indices would contribute. Another reason to extract at deeper layers is that the effective filters operating over image patches in the layers represent relatively 'mature' features with respect to the classification goal, so will tend to simply be closer to $\boldsymbol{g}$. Scaled MC

$$c_k^l = \frac{\boldsymbol{s}_k^l \cdot \boldsymbol{g}}{\sqrt{d_k}} \tag{12}$$

used in [35] may be modified to resolve any scaling issues brought up by down-sampling from pooling layers, which create local feature vectors of different dimensions. Here $\sqrt{d_k}$ represents the dimension of a key for natural language processing in Recurrent Neural Networks (RNNs), and the scaling counteracts the fact that the dot product grows with $\sqrt{d_k}$. Another scaled version is the cosine similarity

$$c_k^l = cosine[\boldsymbol{s}_k^l, \boldsymbol{g}] = \frac{\boldsymbol{s}_k^l \cdot \boldsymbol{g}}{||\boldsymbol{s}_k^l|| \cdot ||\boldsymbol{g}||} \tag{13}$$

as used in [36], which measures the cosine angle between the two vectors. This by definition is a measure of alignment, where the correlation can be interpreted as how parallel the two vectors are.

The scaling factor of the magnitudes ensures high-valued feature vectors are not over-represented. This model is also usually used for RNNs.

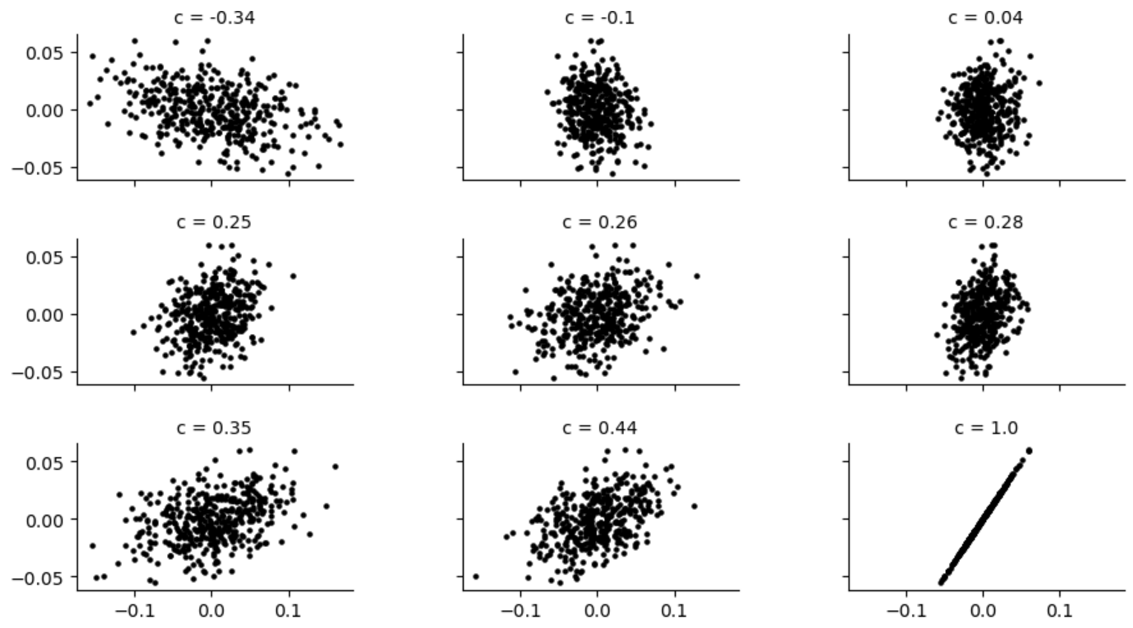

Figure 6: Plots 1 to 9 of a new set of vectors $\{\boldsymbol{v}\} \sim N(\mathbf{0}, \boldsymbol{\Sigma}), |\{\boldsymbol{v}\}| = 9$ against a vector $\boldsymbol{v}_k \in \{\boldsymbol{v}\}, k = 5$ with their Compatibility scores c calculated using Equation 14. Plot 9 is vector $\boldsymbol{v}_k$ against itself

The Spearmans rank correlation coefficient (SRCC) is another widely used measure of correlation, and is defined as PCC between the rank variables [37]. While PCC assesses strictly linear relationships, SRCC assesses monotonic relationships (whether linear or not). A monotonic function preserves or reverses the order between two ordered sets, hence the term 'rank'. If there are no repeated data values, a perfect SRCC of $+1$ or -1 occurs when each of the variables is a perfect monotone function of the other. Not only is SRCC appropriate for both continuous and discrete variables [38], it will likely better fit the often nonlinear feature vectors as long as they are monotonic in nature. As a Compatibility function, we use the discrete form

$$c_k^l = 1 - \frac{6\sum_1^n d_i^2}{n(n^2-1)} \tag{14}$$

where n is the number of ranks, given by the integer length of the feature vectors. We are assuming their dimensionalities have been scaled to equal each other. d_i is difference between the element ranks of the two vectors at index i. SRCC Compatibility is trivially a measure of correlation. Figure 6 shows an improvement over Figure 5, correctly ordering the plots by their strength of correlation, and fixing the range issue. Since $c_k^l \in [-1, 1]$, the exponential weighting will not be as harsh. The maximum magnitudinal difference is only $e^1/e^{-1} \sim 10^1$, thus will yield much softer Attention than general MC above. This could be beneficial for noisy data, where salient regions have broader distributions. SRCC Compatibility is, however, absent in Attention literature. This is likely due to the Attention being, in fact, too soft. Exponential weighting on Feature Maps with hundreds of feature vectors will not be sufficient to highlight salient regions at all; a_k^l for highly correlated $\boldsymbol{s}_k^l, \boldsymbol{g}$ will quickly be drowned out, and produce homogeneous Attention Maps. This could fixed by multiplying all c_k^l by a constant scalar $r > 1$ to extend the range to $[-r, r]$. This adds a data-contingent continuous hyperparameter however, which is unideal for creating a universal CNN that requires no human adjustment to achieve optimality. Figure 7 highlights the stark difference in softness between the SRCC and General MC Attention Maps, with just 9 data points.

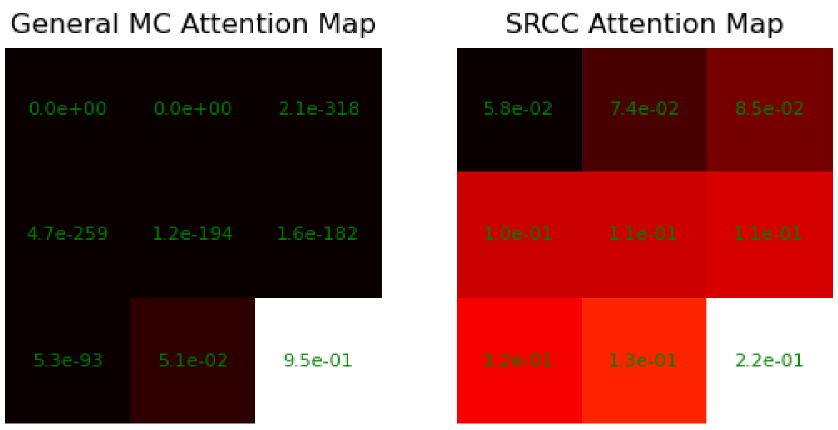

Figure 7: Attention Heat Maps representing the Attention Coefficients of plots 1-9 from Figures 5 (General MC) and 6 (SRCC), calculated using Equation 10. the higher the Attention Coefficient, the lighter the colour.

5.2 Additive Compatibility

The alignment model from [39] can be re-purposed as an AC function given by

$$c_k^l = \boldsymbol{\Psi} \cdot (\boldsymbol{s}_k^l + \boldsymbol{g}) \tag{15}$$

where $\boldsymbol{\Psi} \in \mathbb{R}^{F_l}$ is a learnable parameter, and can be interpreted as learning the universal set of features relevant to the object categories in the dataset. Since free parameters exist between the local and the global image descriptors in a CNN pipeline, we can generalise concatenation of the two descriptors to an addition operation. This allows us to limit the parameters of the Compatibility function. We then learn a single fully connected mapping from the resultant descriptor to the Compatibility scores. In order to introduce more variability for maximum alignment, this can be extended to a more general linear additive model given by

$$c_k^l = \boldsymbol{\Psi} act(\boldsymbol{W_s} \boldsymbol{s}_k^l + \boldsymbol{W_g} \boldsymbol{g} + \boldsymbol{b_g}) + b_\Psi \tag{16}$$

as used in [32], where $\boldsymbol{W_s} \in \mathbb{R}^{F_x \times F_l}, \boldsymbol{W_g} \in \mathbb{R}^{F_x \times F_g}, \boldsymbol{\Psi} \in \mathbb{R}^{C_x}$ and bias terms $b_\Psi \in \mathbb{R}, \boldsymbol{b_g} \in \mathbb{R}^{C_x}$. *act* is a nonlinear activation function, such as ReLU. The benefit of the generalised approach is that introducing $\boldsymbol{W_s}$ allows the deeper fine-scale layers to focus less on generating a signal that is compatible to $\boldsymbol{g}$, which helps it focus on learning the discriminant, diverse features. By also introducing $\boldsymbol{W_g}$ and *act*, the network is allowed to learn nonlinear relationships between the vectors, which is more expressive (see Section 2.1). This is especially important in applications where images have an inherently higher noise to signal ratio, such as Magnetic Resonance Imaging (MRI), where the region of interest is often highly non-homogeneous. A strictly linear AC model may be too sensitive to such fluctuation. For instance, consider the edge detection Kernel $\boldsymbol{K}$ from Equation 9 incorporated in a Convolutional layer to help extract boundaries of brain regions in MRI scans. In a noisy scan, the boundary may be blurred and melted into its surroundings, thus $\boldsymbol{K}$ will fail to fully highlight the boundary edge. MC will interpret this as a non-salient region, however a nonlinear AC model could learn to identify the resulting underlying distribution of pixel values due to $\boldsymbol{K}$ as salient. AC seems appropriate in most if not all case, but is far more computationally expensive than standard MC, since it incorporates an additional learning component. It should therefore be used primarily for low resolution, noisy data like MRIs, while MC is preferable for dense, multi-channel images.

This general model doesn't resemble any common existing measures of correlation. However, it does resemble the function of a single neuron, since it contains a linear function of weights and inputs, followed by a non-linearity. Since the inputs themselves are vectors, it can be interpreted as a single-layer MLP, where the weight matrices are learned via back-propagation. In this case, the Compatibility of the local and global vector isn't derived from a strict, constant method, but instead learned by analysing the final output. Since an arbitrarily large MLP is a universal function approximator [40], a trained single-layer MLP as an AC can be interpreted as an alignment, or dataset specific correlation approximator.

5.3 Significance

There is a mathematical significance of Compatibility representing a measure of correlation rather than pure equivalency. Consider an image piped through a trained Attention-based CNN. Now consider the same image, but with a higher contrast, or exposure, or a similar transformation that changes all the pixel values, but retains features. The raw equivalence between the original and altered will be drastic, and taking a matrix difference will yield a high deviation. However, analysing the pixel mapping will yield a constant relation that can described as a correlation. For instance, every pixel may be multiplied a scalar k, or blurred with a Kernel matrix. Therefore, if two images are considered to be correlated rather than equivalent, we can say they contain the same features. $C(\boldsymbol{\xi}^l, \boldsymbol{g})$ should therefore yield similar Compatibility scores, and thus output the same $\hat{y}_i$. Another implication is that for images with a high signal to noise ratio, where we can look toward other commonly used multiplicative correlation measures to measure our Compatibility; the total number is theoretically infinite.

Moving past pure statistics, the biological significance cannot be understated. Attention is the next logical step in bridging the gap between Artificial Intelligence and Human Intelligence. Understanding what it measures can give us insight into what 'measure' our brain uses to continuously and subconsciously identify salient regions in what it sees.

6 Conclusion

As society becomes more automated, the demand for highly efficient Artificial Intelligence algorithms to process vast amounts of data is ever increasing. CNNs for Computer Vision are just one of many. The implementation of Attention increases efficiency by identifying salient regions of images prior to classification, bearing a striking semblance to human vision. Analysis shows all forms of Multiplicative Compatibility can be interpreted as measures of correlation. Additive Compatibility is a learned process that resembles no existing correlation measure, but through training can be interpreted as learning a data specific alignment, or correlation approximator. MC measure are sensitive to noise, different measures produce vastly different results, but are computationally cheap. AC takes a universal form, handles noisy data well, but is computationally expensive.

References

[1] Utgoff P. and Stracuzzi D. “Many-layered learning”. In: *Neural Computation* 14.10 (2002), pp. 2497–2529.

[2] Schrager J. and Johnson M. “Dynamic plasticity influences the emergence of function in a simple cortical array”. In: *Neural Networks* 9.7 (1996), pp. 1119–1129.

[3] Zewen et al. *A Survey of Convolutional Neural Networks: Analysis, Applications, and Prospects.* 2020, p. 1.

[4] Ward L. *Attention.* URL: `http://www.scholarpedia.org/article/Attention`. (accessed: 02.11.2020).

[5] Walther et al. “On the usefulness of attention for object recognition”. In: *Workshop on Attention and Performance in Computational Vision at ECCV* (2004), pp. 96–103.

[6] Borji et al. “Salient object detection: A benchmark”. In: *IEEE Trans. Image Process* 24.12 (2015), pp. 5706–5722.

[7] Chen et al. “Design and Implementation of Cloud Analytics-Assisted Smart Power Meters Considering Advanced Artificial Intelligence as Edge Analytics in Demand-Side Management for Smart Homes”. In: *Sensors* 2047.19 (2019), p. 3.

[8] Walczaka S. and Narciso C. *Encyclopedia of Physical Science and Technology (Third Edition).* Academic Press, 2001, pp. 631–645.

[9] Lobova et al. “Assessing the anthropogenic impact on Lake Shira from antibiotic resistance of heterotrophic bacteria by neural networks methods”. In: *Microbiology* 76.2 (2007), p. 230.

[10] Baeldung. *Inadequacy of Linear Models: the Road to Nonlinear Functions.* URL: `https://www.baeldung.com/cs/ml-linear-activation-functions`. (accessed: 02.11.2020).

[11] Schmidhuber J. “Deep learning in neural networks: An overview”. In: *Microbiology* 61 (2015), pp. 85–117.

[12] Kumar Dash A. *Feed Forward Neural Networks.* URL: `https://iq.opengenus.org/feed-forward-neural-networks/`. (accessed: 02.11.2020).

[13] Zell A. *Simulation Neuronaler Netze. (German) [Simulation of Neural Networks].* Addison-Wesley, 1994, p. 73.

[14] Stanford CS231n faculty. *CS231n Convolutional Neural Networks for Visual Recognition.* URL: `https://cs231n.github.io/neural-networks-1/`. (accessed: 02.11.2020).

[15] Rumelhart et al. “Learning representations by back-propagating errors”. In: *Nature* 323 (1986), pp. 533–536.

[16] Golik et al. “Cross-Entropy vs. Squared Error Training: a Theoretical and Experimental Comparison”. In: 2013, pp. 1756–1760.

[17] Ioffe S. and Szegedy C. “Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift”. In: vol. 37. Proceedings of Machine Learning Research. PMLR, 2015, pp. 448–456.

[18] Lecun et al. “Efficient BackProp”. In: *Neural Networks: Tricks of the Trade, This Book is an Outgrowth of a 1996 NIPS Workshop.* Berlin, Heidelberg: Springer-Verlag, 1998, pp. 9–50.

[19] Valueva et al. “Application of the residue number system to reduce hardware costs of the convolutional neural network implementation”. In: *Mathematics and Computers in Simulation* 177 (2020), pp. 232–243.

[20] Culurciello E. *Neural Network Architectures.* URL: `https://towardsdatascience.com/neural-network-architectures-156e5bad51ba`. (accessed: 03.11.2020).

[21] Vogel S. *Can a simple CNN work as well as Facial Recognition for differentiating Redheads?* URL: `https://towardsdatascience.com/can-a-simple-cnn-work-as-well-as-facial-recognition-for-differentiating-redheads-18596b05fdec`. (accessed: 03.11.2020).

[22] Sabyasachi S. *Deciding optimal kernel size for CNN.* URL: `https://towardsdatascience.com/deciding-optimal-filter-size-for-cnns-d6f7b56f9363`. (accessed: 09.11.2020).

[23] Zhang et al. *Dive into Deep Learning.* 2020.

[24] Chadha G. and Schwung A. *Learning the Non-linearity in Convolutional Neural Networks.* 2019. URL: `https://arxiv.org/pdf/1905.12337.pdf`.

[25] Yamaguchi et al. “A neural network for speaker-independent isolated word recognition”. In: *ICSLP-1990* (1990), pp. 1077–1080.

[26] Chen et al. “Deep Feature Extraction and Classification of Hyperspectral Images Based on Convolutional Neural Networks”. In: *IEEE Transactions on Geoscience and Remote Sensing* 54.10 (2016), pp. 6232–6251.

[27] Lee et al. “Convolutional Deep Belief Networks for Scalable Unsupervised Learning of Hierarchical Representations”. In: *Proceedings of the 26th Annual International Conference on Machine Learning.* Association for Computing Machinery, 2009, pp. 609–616.

[28] Du et al. “An Improved Quantum-Behaved Particle Swarm Optimization for Endmember Extraction”. In: *IEEE Transactions on Geoscience and Remote Sensing* 57.8 (2019), pp. 6003–6017.

[29] Li et al. “Attention-Guided Unified Network for Panoptic Segmentation”. In: 2019, pp. 7026–7035.

[30] Simonyan et al. “Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps”. In: *CoRR* (2014).

[31] Jetley et al. “Learn to pay attention”. In: *International Conference on Learning Representations (2018)* 124 (2018), pp. 117–12.

[32] Schlemper et al. “Attention-Gated Networks for Improving Ultrasound Scan Plane Detection”. In: *International Conference on Medical Image Computing and Computer-Assisted Intervention* (2018).

[33] Luong et al. “Effective Approaches to Attention-based Neural Machine Translation”. In: *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing.* Association for Computational Linguistics, Sept. 2015, pp. 1412–1421.

[34] Pearson K. “Notes on regression and inheritance in the case of two parents”. In: *Proceedings of the Royal Society of London* 58 (1895), pp. 240–242.

[35] Vaswani et al. “Attention is All You Need”. In: *Proceedings of the 31st International Conference on Neural Information Processing Systems*. Curran Associates Inc., 2017, pp. 6000–6010.

[36] Graves et al. *Neural Turing Machines*. 2014. URL: `http://arxiv.org/abs/1410.5401`.

[37] Myers et al. *Research Design and Statistical Analysis*. 2nd ed. Lawrence Erlbaum, 2003, p. 508.

[38] Lehman et al. *JMP for Basic Univariate and Multivariate Statistics: A Step-by-Step Guide*. 2005, p. 123.

[39] Bahdanau et al. *Neural Machine Translation by Jointly Learning to Align and Translate*. 2014. URL: `http://arxiv.org/abs/1409.0473`.

[40] Sifaoui et al. “On the Use of Neural Network as a Universal Approximator”. In: *International Journal of Sciences* 2 (2008), pp. 386–399.