Informational Image Data Preprocessing: IFE Blood Samples

Mert Saruhan

Supervisor: Prof. Dr. rer. nat. habil. Thomas Villmann

April 3, 2023

Abstract

In this paper, we approach the supervised classification of image data with a different perspective than the normalized way in the data science sector. Typically, it is a norm for data scientists in the industry to use Neural Networks to train machines for nearly any task. The preference for using Neural Networks comes from the high prediction power of the model. However, the primary drawback of the Neural Network is that we cannot always understand the logic behind the hidden layers of the neural network models. That is why in this paper, we tried to approach image data by using structured data for machine learning. The image data we can use for this purpose contains information rather than any image. The data we use is blood tests with labels. This paper primarily focuses on analyzing the SP bar of IFE samples and the image processing phase, which involves transforming the data into a format suitable for training on LVQ models. We hope one can provide a clearer understanding of the decision-making process using LVQ methods. The result from LVQ methods might offer a viable alternative to Neural Networks for some image classification tasks.

Keywords: LVQ, Neural Network, machine learning, pre-processing, IFE, immunofixation, electrophoresis

Contents

1 Introduction

The main object of this paper is to transform image data into structured data, which has informational data such as bars and graphs. The norm for using image data in machine learning is creating a Convolutional Neural Network (CNN) on the image data's pixel values. The convolution part of CNN takes the pixel values of image data and first convolute them with a kernel some number of times. Convolution here changes the image size and values of the pixels. After every convolution, we do pooling. Pooling can also make the image smaller. These convolutions and pooling back-to-back make the model learn better from data. After the convolution phase, we create a Neural Network, where the input of the Neural Network is the convoluted pixel values.

As we summarize the CNN here, we can see the CNN method uses pixel values to learn instead of the present information already on those images. We really cannot understand how CNN learns from these images since CNN does not read the images but instead groups the images by shapes. Our task here is to extract the information on the image data into structured data to use in machine learning models. This way, we would better understand how our model works instead of relying on Neural Network, which has difficulty interpreting the logic behind the hidden layers.

Our paper uses blood test data, specifically Immunofixation Electrophoresis (IFE) data. IFE is a method to read albumin and globulin (also called immunoglobulin) in blood. According to the article on MedlinePlus[1], measuring these proteins can help diagnose various diseases.

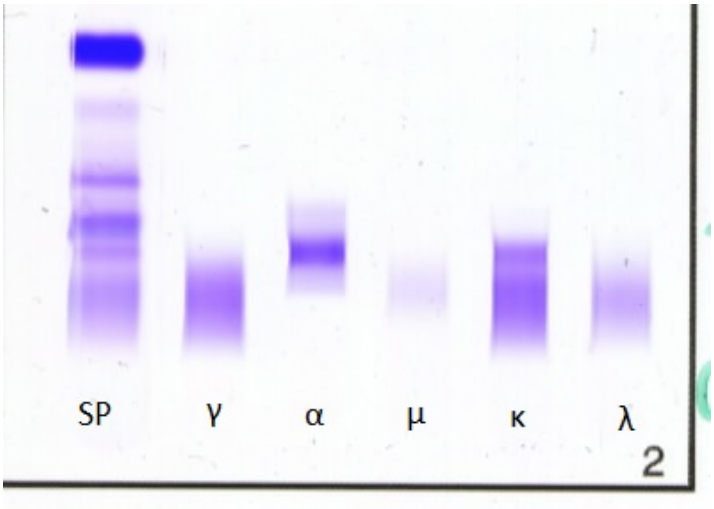
IFE is used to detect problems such as:

- Help in the diagnose and monitoring of lymphoma, chronic lymphocytic leukemia or monoclonal gammopathies, such as multiple myeloma
- Investigate abnormal findings on other laboratory tests, such as total protein, albumin level, elevated calcium levels, or low white or red blood cell counts
- Evaluate someone for an inflammatory condition, an autoimmune disease, an infection, a kidney or liver disorder

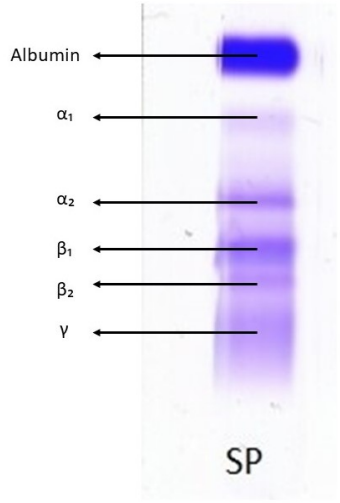
The list is taken from Testing.com [2].

According to Tuve et al. [3], the IFE results have 6 bars, namely Serum Protein Electrophoresis (SP) or Marker, γ, α, μ, κ, and λ in that order. As we can see from the Figure 1a, the SP bar has better borders than other bars in the IFE results. Wahed [4] states SP has six bands, from top to bottom, named: albumin, α_1, α_2, β_1, β_2, and γ, as we can see from the Figure 1b. We check these bands to understand whether the patient has an abnormal blood test. However, even though we can understand there is an abnormality with the blood test, we cannot understand the problem because the SP bar does not give us which immunoglobulin is causing the trouble. According to Leung, the reason is that SP measures the proteins in the blood quantitatively; however, the IFE test can give us the qualitative results of the blood protein, which means which protein other than albumin is abundant in the blood sample[5]. Unlike the SP bar, the IFE test gives us tests of three heavy protein chains; Gamma (γ), Alpha (α), and Mu (μ), and two light protein chains; Kappa (κ) and Lambda (λ). Heavy and light protein chains together construct the immunoglobulins. For example, IgM-Kappa means in

blood immunoglobulin μ with light protein chain κ detected more than other proteins. We can see the structure in Figure 2.



(a) Our sample IFE image.



(b) Our sample image's SP part. These band name positions are just representative to show the order of the band names, and should not be taken as exact correct places for the bands.

Figure 1: IFE and SP images

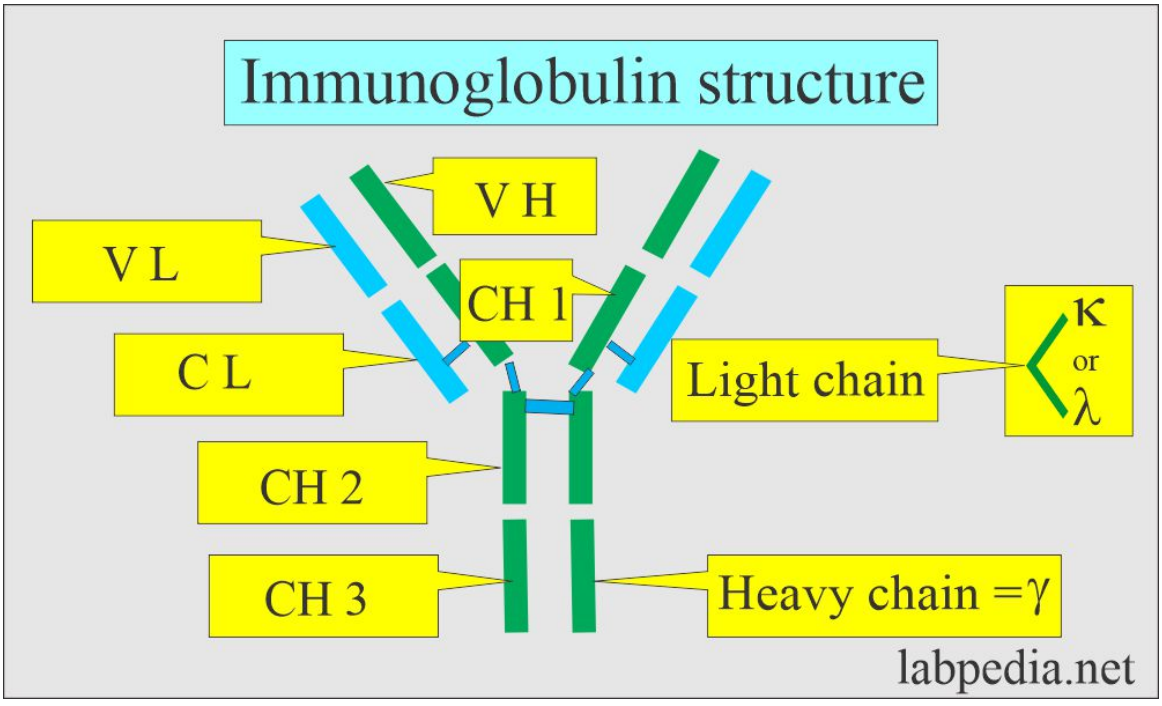


Figure 2: Immunoglobulin structure, image is taken from labpedia.net [6].

We mentioned two types of α and γ. One type is the band that is observed on the SP bars and the other type is the bars on the IFE test. These can be confusing, so be careful when reading this paper. In this paper we call bands on SP bars as bands and bars on IFE test as bars.

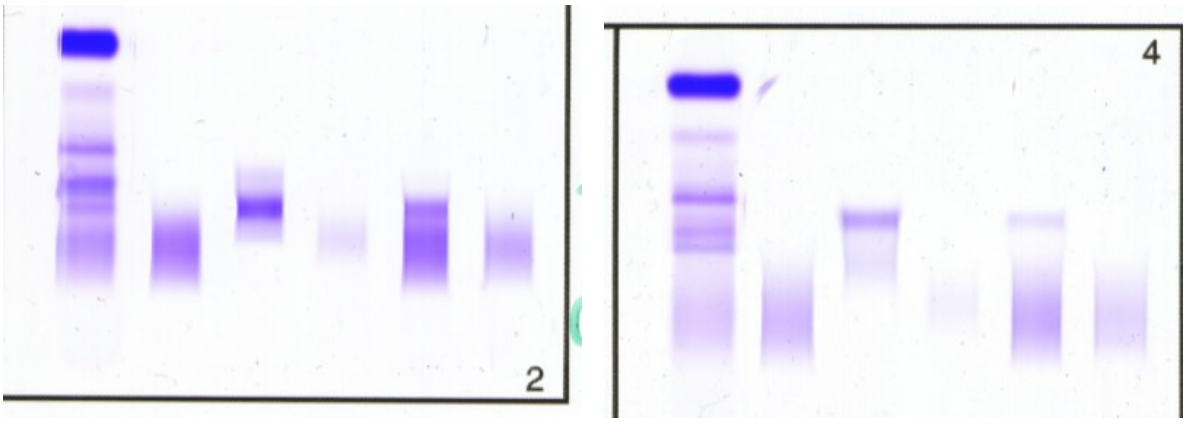
According to an article on Eclinpath, the IFE method divides the proteins by their charge and size, where albumin has the most negative charge among other proteins in the SP bar [7]. Since albumin is the most abundant protein in the SP bar, according to Leung, it shows a single thick band on SP bar [5]. Unfortunately, we could not find any information about the distances between bands in electrophoresis tests since the densities of the bands are essential to diagnose than the position. Having band distances would be helpful for our paper. Still, instead of finding a reference, we will analyze the SP bars and try to find the average distances between the SP bands with statistics.

2 Method

2.1 Image Reading

To read the image, we use two libraries, namely scikit-image, and Pillow. First, to read the image pixels, we turn the image into NumPy 2d array with the array function in NumPy. Since the image is colored, it returns a 2d array with three channels. Then after turning our image into a readable array, we find the borders with scikit-image function. Scikit-image has a function called filters.sobel, which finds the edges of the given image. The function takes an image array as input and returns a new image array, which in the new array the image has brighter edges on a black background. The function here uses a convolution kernel to find the images, and one can find more information about the function at scikit-image.org. One can do a similar job by writing a convolution function from zero. However, we choose here to use a pre-written function. After finding the boundaries, we turn it back to an image since we use three channels which are RGB values and want to use black and white images. To turn the edge image into an array, we use the fromarray function from Pillow. While doing this process, we multiply the output of the edge matrix to thicken the edges more before turning it into an RGB image. Now that the edges are highlighted, we can transform the image into black and white and then transform it back into a numpy array with Pillow. Since we thickened the bars and highlighted them compared to the background, it would be easier to find the bars' position. So now we can start finding the bars' position starting with the SP bar.

For this paper, as an example we use one of the images in Figure 3 as test image and the other one as the reference image to automate the bar finding process. We use the test image for find bar positions and values and reference image for creating the albumin mask, taking a reference length for the bars, and finding the distances between the bars in each image. One can use different reference images for each of the tasks that require reference image, but for simplicity, in this paper we use one image as a reference image.



(a) Test image (b) Reference image

Figure 3: Images we use for this paper.

More information about Pillow and scikit-image:
Pillow: https://pypi.org/project/Pillow/
Scikit-image: https://scikit-image.org/

2.2 Finding The SP Bar

2.2.1 Finding the albumin mask

To find the SP bar, we use a convolution mask. We create an albumin-sized matrix that finds the albumin's position in the image and indicates the SP bar's position. However, we are going to talk about this later in Subsection 2.2.2.

To create an albumin-sized matrix, we first need to find the size of the albumin from the SP bar of one of the images. We named the image albumin reference image to find other images efficiently. For that, we first need to find the leftmost pixel, which has a bigger value than background values (since we use a black background and white colored borders). We start with the left pixel since it is easier to find than the top pixel. The reason is that when we add the column values, the columns values of the bar mostly read the bar section rather than the background. Suppose we would first try to find the top or bottom index. In that case, we need to take the row mean, and that would give us a smaller variance in the row mean values, and that would be harder to distinguish the background and the bar.

To find the albumin's left index, we take the mean value of all the columns of the albumin reference image as we can see on Figure 4. For our reference image, we also take the median value of the column mean values, which gives us the border value of the background and the bar values. We can amplify the border with some number to make it more precise. We name the amplified border value as the threshold and now look for the values where the mean of column values passes the threshold.

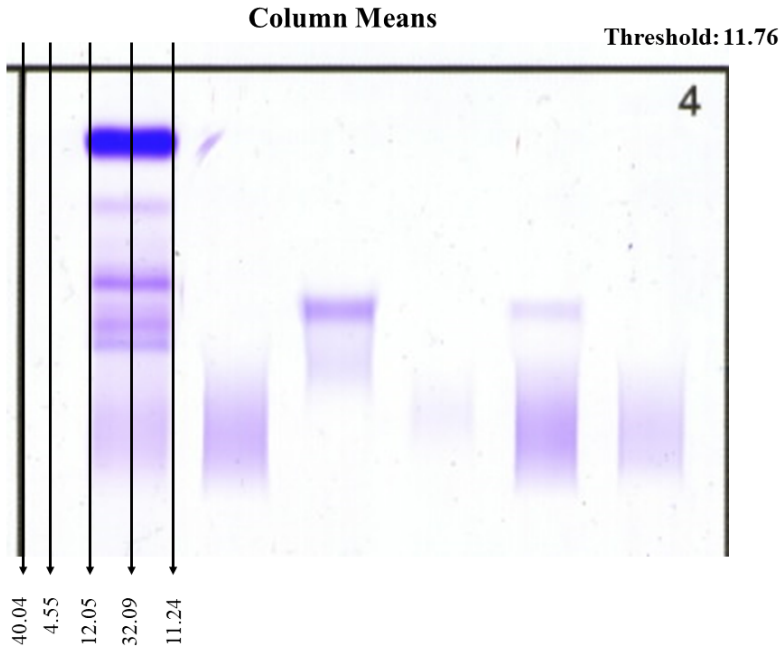


Figure 4: Taking means of all of the column pixel values of the reference image.

Then we look at the column mean values from left to right. We are interested in the values bigger than the threshold we specified earlier, but there is a catch. Since our images contain some borders around the image, our search would also detect them. We are not interested in the image borders, so we need to define a thickness for our bars. The thickness should be bigger than the image border pixel width and smaller than our bar pixel width. After finding

a good fit for the thickness pixel value, we can start searching the leftmost pixel of the bar. We look at column means that are greater than the threshold value and continue to be greater than the threshold until the length of our thickness value. If we can find such an index, we take the starting index as our left index of the bar. Suppose we cannot find such a value. In that case, we need to decrease either the thickness or the threshold by changing the thickness value or threshold amplifier, respectively.

Since we found the left index, we can now find the right index of the SP bar similarly. Starting from the *left index+thickness*, since we know from the left index to the thickness value right, all the values are greater than the threshold. We still look at the column mean values of the image. Where the value gets less than our threshold will be our right index for our SP image, which will also be the right index for our albumin mask.

To find the albumin's top and bottom, we are going to look at the row mean values, but first, we need to cut out the SP bar from the whole image. The reason is, as we have discussed earlier, the row mean values do not have enough variance for us to distinguish the background and the bar. If we will not cut the image, then the background pixels would take the mean of the row values close to the threshold value. That is why we first cut the image from the left and right index, so there will not be any background noise to disturb the row mean. Here we do not need to change the threshold value since the threshold still distinguishes the bars' pixels from the background. Suppose we would calculate the threshold after cutting the image. In that case, our new cut image contains mostly white pixels since the bar contains most of the image. The new median would be closer to the white pixel value, which might give bad results. We have a working threshold value, and it is unnecessary to recalculate it. For thickness, if we had chosen a good value that does not exceed the height of the albumin, we could still use it. Otherwise, the thickness value rejects the albumin bar, and we need to lower the thickness value. The best would be choosing a thickness value that would be smaller than the dimensions of the albumin and greater than the thickness of the image border. We can see the cut reference image and the mean row values in Figure 5.

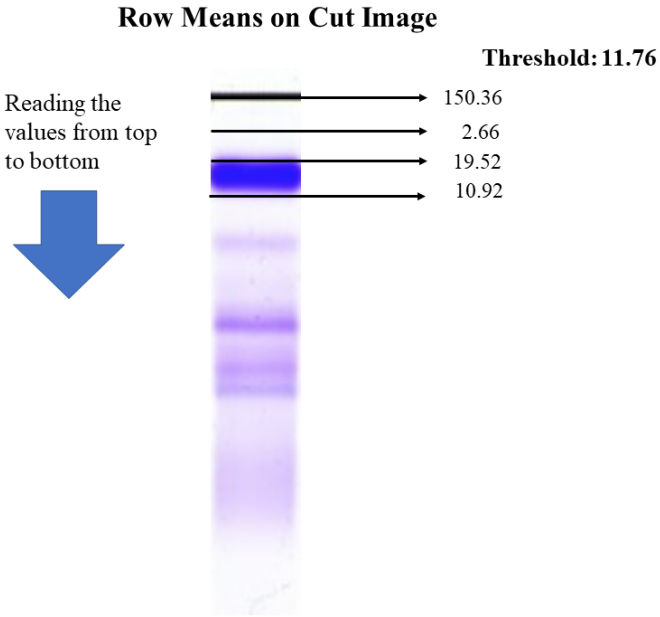


Figure 5: Reading row mean values of the reference image to detect the albumin band.

After cutting the image, we used the same way we found the left and right indices. We look at every row mean value from top to bottom. For every index, we seek the thickness value of many row values which exceed our threshold for the thickness value we chose earlier. After finding such an index, we name it the top index of our albumin cutout. Then we find the bottom of the albumin cutout as seeking the row value, which comes after the top index and less than the threshold value. We seek from the *top row+thickness value* to the bottom until we come across a value lower than the threshold.

In the end, we have indices for the left, right, and top of the SP bar, which are also the same indices for the albumin and bottom of the albumin index. Then take the difference between the left and right indices to find the width and the difference between the top and bottom indices to find the height of the albumin cutout. Here we can also make the size of the albumin cutout smaller to increase the precision of detecting the albumins on other images.

2.2.2 Finding SP bars with the albumin mask

Since now we have dimensions of how albumin looks like, we can create a mask out of it and search for the albumins in all the images. For that, first, we create a matrix (kernel) where the size is the same as the albumin's width and height, and the values of the matrix are all 1. One can change the values of the matrix to create different convolutions. However, in the code, we used ones for all the values, indicating that we sum all the values of the given area. One can give the outer part of the matrix lower or higher values to create a better center or border detection, respectively.

After creating the mask, we convolute the matrix with the image. Where the convolution gives the maximum value, it indicates the albumin. We wait for convolution to give the maximum value at the albumin since the albumin's brightness (pixel value) is the biggest among the other parts of the bars. The only parts in the image brighter than the albumin pixels are the pixels of the image borders. However, they do not get the maximum value since they are thin and cannot fill the albumin mask. Therefore, albumin detection works.

Another point to mention is how to run albumin detection faster. We do not need to read all the images since albumin is available at the top-left quarter of the image. So, we can only use our convolution method to pinpoint the albumin hence the SP bar's position. Also, we can skip a small number of pixels either right or down (depending on the position of the mask at that time) to make the process less accurate but faster. In Figure 6 we can see how the process goes visually.

At the end of the process, we detect the left and top indices of the given image's albumin. We assume the top of the albumin is the top of the bar, and the left of the albumin is left of the bar. Since we know how wide the albumin is from our albumin mask and accepting all the images have similar size and ratio aspects, we can also find the right index of the SP bar just by adding the width of the albumin mask to the left index. Therefore, we found the left, right, and top of the SP bar, and our next step would be finding the bottom of the SP bar.

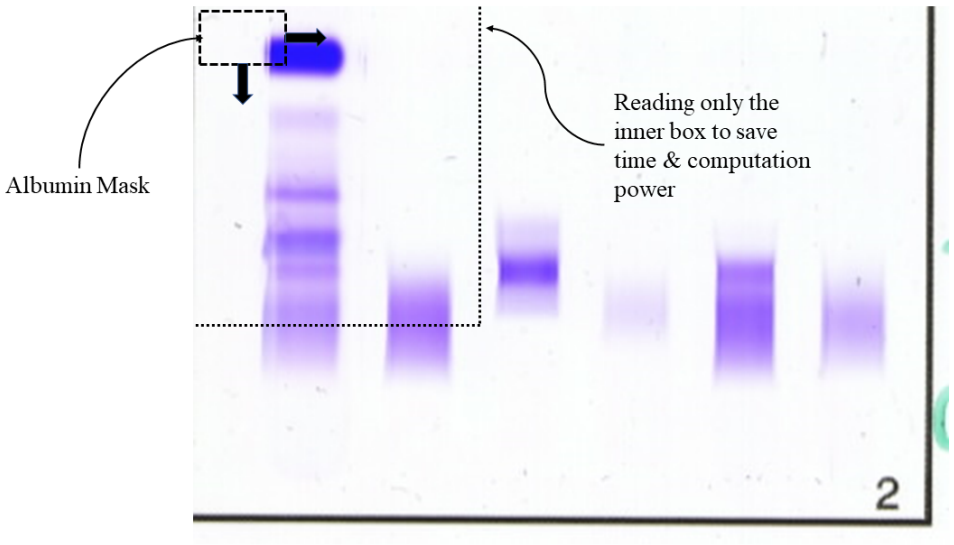


Figure 6: Finding mask's maximum value to detect albumin on test image.

2.2.3 Finding length

Since the bars' lengths are similar, we only need to find one of the SP bar's lengths, and then we can implement that length to other images. For that, we can use our albumin reference image or pick another reference in which we can read the length better than the prior.

First, we need to find the SP bar itself to find the bottom index of a reference bar. For that, we use our albumin convolution method to find the position that we described earlier in Subsection 2.2.2. The albumin convolution method gives us the SP bar's left, right, and top indices. Now we cut the image out by using the indices we found and taking the row mean of the created image matrix.

We use a similar method we used earlier when finding the albumin reference. However, we start reading the row mean values from bottom to top this time. We still use the threshold we chose earlier in Subsection 2.2.1 to distinguish the bar and background. However, suppose we use another image to find the height of the bar than we used for albumin reference. In that case, we need to find a new threshold for our new image since the threshold is related to a median of column mean values, and the value is slightly different for all images.

Some images have ink stains since they are handmade and might cause problems. The best way to avoid such stains is just by selecting a reference image for height with no stains. Nevertheless, suppose we do not have a chance. In that case, we can divide the row mean values into two, namely left row means and right row means, and seek our bottom bar by looking at both values together as seen on the Figure 7. To choose the bottom index, we read left and right means row values from bottom to top and seek an index where for n number of times, both row mean values are greater than our threshold for that image. After finding the bottom index of the SP bar, we take the index number as our reference height since we have already cut out the top part of the image. Now we can use reference height on the other images to find the whole SP bar.

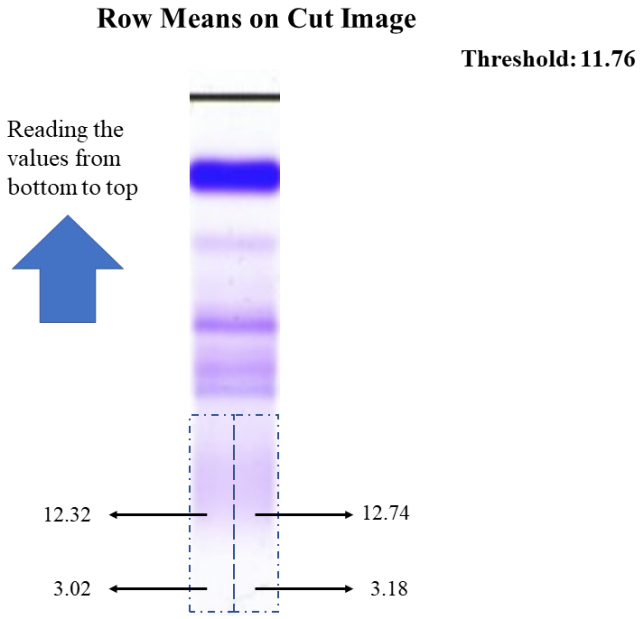


Figure 7: Finding bottom index of the bar.

Now we have two functions working together to find the SP bar for any image. First, we detect the albumin position on any given image by using convolution on the albumin mask. It gives us the SP bar's left, right, and top indices. After finding these three indices, we can find the bottom index by adding reference height to the top index. Moreover, that is how we find the position of the SP bar on any given IFE image.

2.2.4 Finding other bars

Since we have our SP bar for all our images, we can now start finding other bars to read the values. Except for the SP bar, we have five more bars left, namely; γ, α, μ, κ, and λ, from left to right in that order. Again, here we assume all the images are similar, and the distances of the bars in our images are equal. If our images had different sizes or shapes than each other, then with our code, it would be harder to detect the bars.

Since the distances of the bars are equal, we can find only the distance between the SP bar and γ for one image, and we can step that distance to find other bars in that image and others. At least, that is what we thought until we figured out it did not work. If it does not work, we can find all the distances between the bars and use that distance list to find the bars on other images, which is how we approached the solution. So first, we will talk about our bad ways of finding the distance, and then we will explain how to find the distance better for all the images.

2.2.4.1 Bad way #1: Taking only the distance between SP and γ bar

If we try to find the distance between SP and γ bar and assume the distance between any bar is the same, and hope to find the next bar by just shifting the same distance from the previous one seems intuitive. We find the distance from SP to γ bar using the same method we used to find albumin's left index. We look for column mean values of the image starting from the right index of the SP bar and seek an index where the values are above the threshold

for certain many times. Here we also need to use a threshold for every image, which is an extra calculation. After finding such a left index for the γ bar on our image, we can add bar width (which is also the albumin reference's width) to our distance since the distance is from the right side of the SP bar to the left side of the γ bar. Now the calculated number is our distance reference, which is the distance for all bars' left index to the left index of the prior bar.

After calculating the distance and trying to find the other bars in the given image, we found out it does not work. But, what was the problem? Since any small error in the distance can fold up to 5 times until the λ bar, we should not make any error. Otherwise, that makes a great error in reading the last bar. Nevertheless, it was inevitable. If we remember, we used a mask extracted from one image to find the albumin for every image. The drawback is that not all SP bar (hence albumin) has the same width. So, when we detect the albumin, we do not centralize the SP bar perfectly. Also, we use a threshold to find the γ bar's left index, which we can detect the index a little far left or right depending on the threshold value. Since we have not perfected detecting the correct size of albumin or have any correct way to detect where the bar starts and ends, we cannot find the real distance of bars for all images.

2.2.4.2 Bad way #2: Taking the distance between every bar with value reading

After concluding that any small error in distance calculation between two bars might result in a huge error in finding the bar position, we approached the problem by trying to find all the bar distances for one image. Then the distance list would work for other images since the machine that gives the results is the same machine, which means images have the same ratio and size.

To find the distance between any two consecutive bars, we use the same method to find the γ bar, but this time for all the bars. Since we know the SP bar's position, we start with that. From the SP bar's right, we read the column mean values until we find the repetitive values, which pass the threshold value. Then we name the first consecutive indices as γ bar's left index. After finding the left index of the γ bar, we can easily calculate the right index of γ bar by just adding the bar width, which is determined by our albumin mask. Then we repeat the same process for μ, starting from γ bar's right index. We repeat the same process until finding the left index of λ. After finding all the left indices of the bars in one image, we can take the left index distances of the consecutive bars. For other images, we can add the distance from the left of the prior bar to find the left index of the next bar, and by adding the width of the bar, we can cut the bar.

The problem with this method is that the column mean-seeking method is useless for many images. Except for the SP bar, all the other bars have dimmer colors. The low pixel value makes the column mean values lower than our threshold since most of the image is the background. So, if we miss even just one bar, the code will confuse the positions of the bars, misname them, and can only read at most 5 bars. So, we need a better way to find the bar's position.

2.2.4.3 A good way: Using bar mask

By learning from our mistakes and thinking about our older methods, we created a bar mask. We know what our bars look like, or at least we know the approximate dimensions. All the bars have the same dimensions as the SP bar and are on the same line (assuming that

the image is not tilted). As we can see from Figure 1a in the introduction, the SP bar covers more spectrum than other bars. So we can take the dimensions of the SP bar and create a mask out of it for one reference image and search the other bars horizontally by using the mask.

We create our bar mask the same way before; all the matrix values are 1. Then we multiply the matrix, starting from the right index of the SP bar and the same top and bottom position of the SP bar. The important step to be cautious of here would be that we should not go all the way read all the image to find the bars. We still need to know how far the next bar is from the first one with respect to the image width. If we run our bar mask more than we should, we might find the next bar or a space between two bars since our convolution might be getting its maximum value there. We can see the application on the Figure 8.

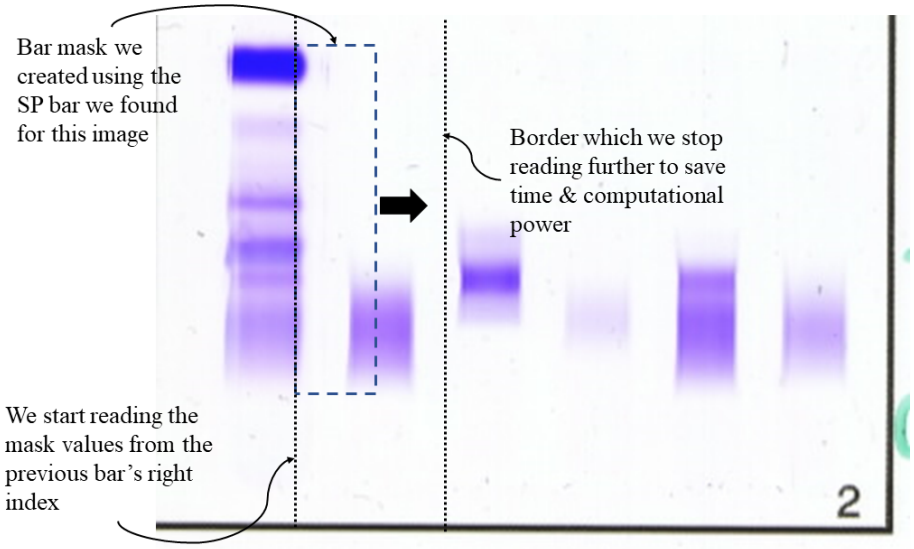


Figure 8: Using detected SP bar as a mask to find other bars. This is an example for finding the γ bar. After finding the γ bar, we are relocating the border further.

Since we started from the SP bar, the position we found our maximum value is the position of our γ bar. Then we take its left index and calculate the distance between the two bars as the difference between their left indices. From here, we can easily find the right index of the γ bar by adding the bar width. We need the right index since we do the same process for the next bar, the μ bar, starting from the right index of the γ bar. After finding all the distances between bars, we can store these distances and find the bars on the other images with this distance reference.

Since not all bars have same width, and we are also having some padding on our bar mask; The bar cutouts are not perfect. The method can still find the approximate position of the bars, but in some bar cutouts, there might be white space around the bar. However, the padding does not affect us to extract the bar values.

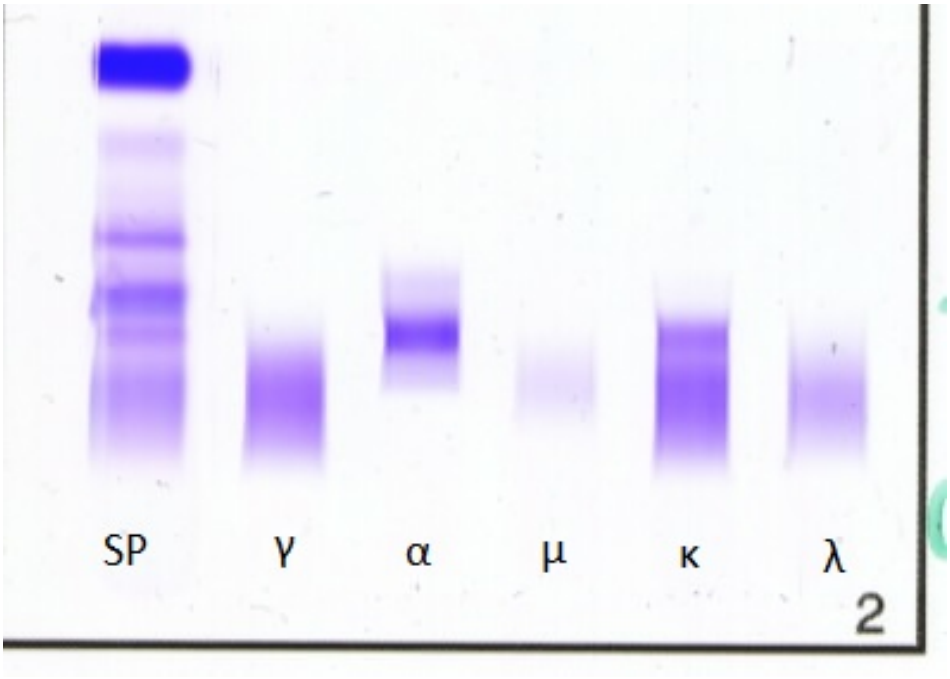
2.3 Reading Bar Values

Now with our bar-finding method, we can approximately and accurately find all the bars in our sample IFE image. Now we need to read the values of the bars for further calculation. We used our manipulated images until here to find the bars' position since the boundaries and values differ more than the original images. However, suppose we try to read the values from our manipulated images. In that case, it is undeniable that the values would be wrong. That is why we save the positions of the bars we found with manipulated images and use them on matrix values of original grayscale images.

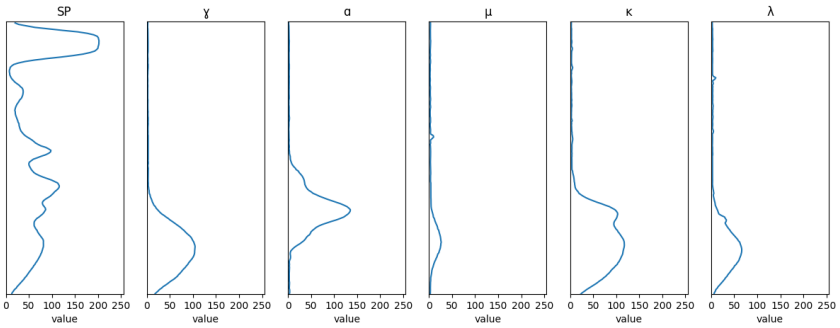
As we mentioned in the above Subsection 2.2.4.3, the widths of the bars are not perfectly matched with our bar cutout. Since reading the background values for the images would create an error in bar values, we need to eliminate the background in our readings. Bar detection is good at centralizing the image. So, we can take the middle point of the image and give the same small radius around it with certainty to stay in the bar and read the mean values of the rows as our bar values for each bar. This process gives us the graph of the bar.

We were using black background because of the manipulation. However, now the original grayscale images have darker bars on white background. So, after calculating the graph of the bars, we need to adjust the values. The matrix of the grayscale images has values between 0 and 255, where 0 is pure black, 255 is white, and everything in between is shades of gray, depending on their value. To get the true graphs, we need to subtract 255 from every value on the graph we found from the bar images. Now our graph values are ready for further analysis.

For visual purposes, we compare colored (original) test image with the values we extracted from grayscale version in Figure 9.



(a) Test IFE image.



(b) Graphs, created by reading the grayscale test image's bars.

Figure 9: Extracting the image values into structured data.

3 Analysis

Until here, we found each bar in all the images, and then we read their values to find the graph of each bar in the images. Now, the data we have turned into structured data can be used with machine learning methods, such as LVQ. However, we will use these graphs to analyze the relationship between the bands of the SP bar. For that, we need to find the lines, which are the middle point of those bands. We use five different methods to find such lines.

3.1 Functions

3.1.1 Derivative

Since we know that where the derivative value is close to 0, that point is a saddle point. We are interested in the local maximums of the values since the local maximum indicates the brightest point of the band, which can be interpretable for the middle point of the band. Of course, in that assumption, we need to say, it does not work for the γ band since it is broad and has an uneven value distribution. So even if we find a middle point for γ bands, we cannot say it is the middle point of the γ band. However, we can use the information to confirm that we found five lines on the bar, which we will take as the reading went successful.

To calculate the derivative of the SP bar, we take the graph values we calculated earlier and subtract every value graph with the previous value. This principle is how we calculate the derivative of any function.

$$D = lim_{a \to b} \frac{f(b) - f(a)}{b - a} \tag{1}$$

Here in Equation (1) D is the derivative of the function at point b. We are not taking the limit, but taking a as $b - 1$. The answer when we take inputs $a = b - 1$ would give us the derivative between $b - 1$ and b, which would be an approximate value of the derivative at point b.

After finding the derivative list, we look at the values to find the peak points. To find the peak point, we look for the point where it is close to 0 from the positive side, and after there, it decreases to a negative value. We have different methods to find such points, but all the following methods use the same derivative method to find the peak points of the graph values.

3.1.2 Sharp-v1

The first method we are going to introduce is the most basic one of all. It is named Sharp-v1 because it can detect sharp changes, and it is the first version of the latter ones. Sharp-v1 function starting from the top of the graph values (reads from albumin to γ band direction) looks at the derivative values, where the values are near 0 and positive. After that, the function looks at the first negative value and takes that position as a line. The function finds any candidate for middle points for the bands. However, the drawback is that it can also find saddle points or small fluctuations in the graph derivative. So, the method is not perfect, but it is still useful. Since other functions are too specific in finding the lines, sometimes they might miss small changes. Sharp-v1 is a great solution to find missed lines at those moments.

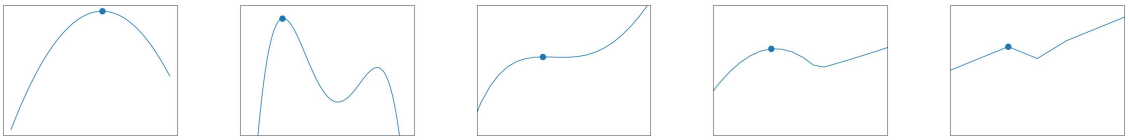


Figure 10: Validated and invalidated function peaks by Sharp-v1 method

3.1.3 Sharp-v2

Sharp-v2 works like Sharp-v1, as one can understand from the name. The method starts from the top of the graph and looks for values where it starts to go negative, but this time we look for the positive and negative values to be next to each other. It is named Sharp-v2 because it is the successor of Sharp-v1. One might wait for Sharp-v2 to find similar results as Sharp-v1, which we have waited for, but Sharp-v2 has a lower success rate than Sharp-v1 to validate the images seen on Table 1.

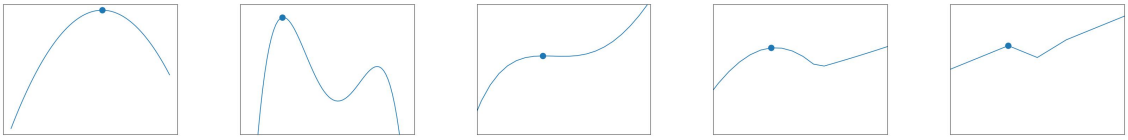


Figure 11: Validated and invalidated function peaks by Sharp-v2 method

3.1.4 Sharp-v3

The last version of the Sharp series is Sharp-v3. Like Sharp-v2, Sharp-v3 uses a relation between two derivative values. First, the method seeks a derivative value from top to bottom, where the value is between 0 and +1. Then it looks at the following value, and if the value decreases to a value between -1 and 0, we take the index as the middle point of the band. All the Sharp versions detect near similar lines, where Sharp-v1 is the winner in validating the greatest number of images because of its non-limitations as also can be seen on Table 1. However, the method validates bars better with the cost of reliability of those lines.

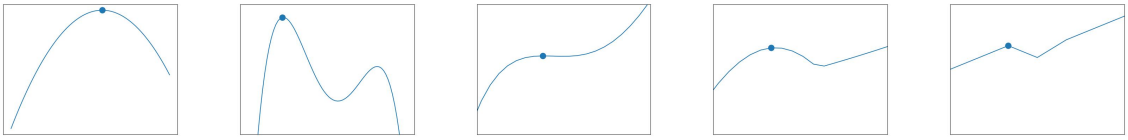


Figure 12: Validated and invalidated function peaks by Sharp-v3 method

3.1.5 Smooth

The Smooth method is the first function we discuss with a different method than the Sharp versions. Smooth method's name comes from the functions it can detect, like Sharp functions. It can detect the peaks where it smoothly increases and decreases without sudden decreases near the peak. As a result, the lines from the Smooth method are peak points of a hill. Unlike Sharp versions, the Smooth function looks for sequential differentiation values.

First, the function detects a point where the derivative value is positive and near 0. We can also say the value is between 0 and +1. Then we look for the derivative values which come after the starting point. Suppose the values after the starting point are all negative until some many times (let us call it n many times), which means the graph is monotonically decreasing for n many times. In that case, it passes the first test. If the point cannot pass the first test, there is no need to check the second test for the given point, and we can start our search from the end of the failed position.

If the function passes the first test, then this time, we look for the values before the starting points as much as we looked for the first test. This time we want the values to be positive since values being positive indicates that the graph is monotonically increasing at that point. Suppose it fails by finding a negative point before our starting point. In that case, we can start searching for a new starting point from that *negative-point*+n since all the points between the failure point and the *point*+n are doomed to fail because of the same negative point.

If the function passes both tests, we can choose the starting point as the peak point. The drawback of this method is that we cannot be sure about the symmetricity of the hill. We assume the bands are symmetric and have peak moments in their middle part. One can improve, or mostly limit, the function by giving a bound for the increase and decrease speed. The graphs do not need to change in constant speed, and one can add acceleration to the change in derivative to limit the peak output numbers. The method would decrease the number of lines found and increase the reliability of finding peak points.

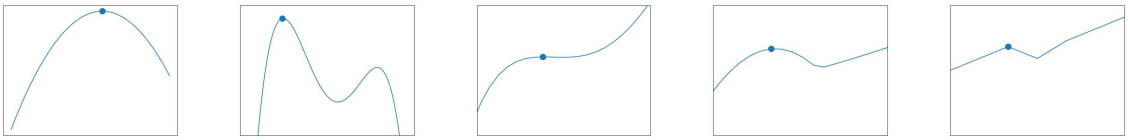


Figure 13: Validated and invalidated function peaks by Smooth method

3.1.6 Dent

Lastly, we have the Dent method. The main part that makes this method unique from the Smooth method is that it ignores small dents near the peak points, which is why we call it the Dent method. It can detect the local peak points of the graphs where the peak point has a small dent between the two peak points close to each other. The method works sequentially like the Dent method, which means we look for a group of derivative values to determine the peak point. The Dent method also covers the peak points that the Smooth method finds. Therefore, we see the Dent method as more inclusive than the Smooth method.

With this method, first, we find a starting point where the derivative is positive and near 0. After, we look n many consecutive left and right derivative values next to our starting point.

Then take their mean values and check the mean values sign. If the sign for the left mean is positive, and the sign for the right mean is negative, then we take the point as a peak point.

Small note: As mentioned earlier, this method can only accept small decreases in n-neighborhood between two peaks. So, a drastic decrease would probably be skipped by this method. Surprisingly, the Dent method has the best performance among the other methods we used.

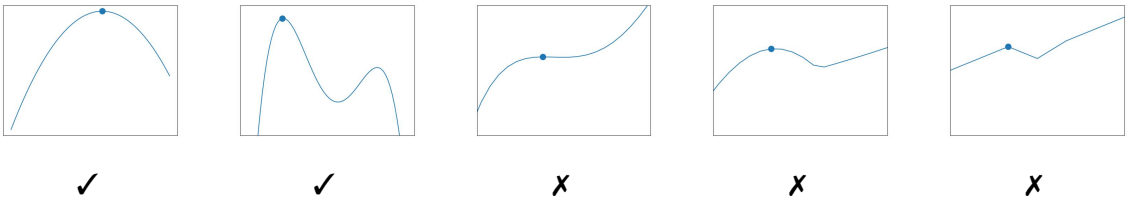


Figure 14: Validated and invalidated function peaks by Dent method

3.2 Line Cleaner

SP bar has 5 to 6 main bands, which are albumin, α_1, α_2, β_1, β_2, and γ bands in that order. According to Leung, sometimes β_2 is available in the SP bar, and sometimes not [5]. While it is available, β_2 is close to β_1. To eliminate this confusion, we can try to count only the main bands except for β_2. Also, sometimes our methods, or a combination of methods which we will mention later, might give two close lines as the center of a band if the center has two peaks near each other. In that case, we would like to eliminate those extra lines. For this purpose, we use the Line Cleaner method.

The Line Cleaner method is simple. We check all the lines, and if any of those lines are in some n-neighborhood to each other, we take their mean position as the main line. In this way, we eliminate the close lines for running to be the same band's middle point and eliminate the β_2 problem by just combining it with β_1. We can see an example of how Line Cleaner works in Figure 15.

We should not use a big neighborhood variable for the lines; if we do so, we might combine the middle points of different bands into one. One way to choose the neighborhood value is picking with hand while assuring that the distance does not exceed half the distance of any bands on the SP bar. Alternatively, we can directly choose half of the albumin's height since the size of the albumin is not too wide to violate the distance between any two bands.

Ultimately, we would like to find five lines, namely albumin, α_1, α_2, β, and γ band lines, which indicate that the finding line methods worked well on the given image.

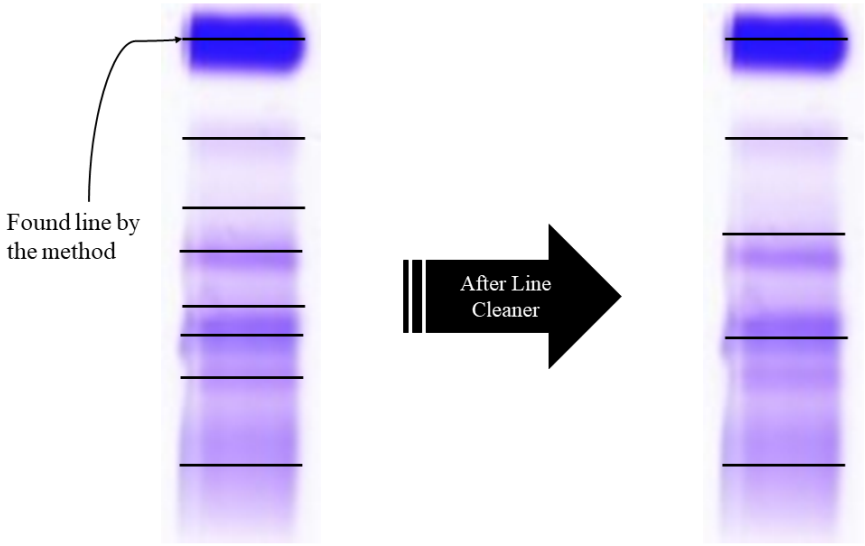


Figure 15: An example of how Line Cleaner works.

3.3 Method Combinations

Now we come to the last step to find the lines. As mentioned, we have five main lines, albumin, α_1, α_2, β, and γ. We accept that we found all the lines correctly and call the image-validated image if we found exactly five lines in the image after using Line Cleaner. Note that we cannot give reliability that we found the middle points correctly. We just assumed the SP graph's peak points indicate the bands' middle points.

Suppose we try to run all the methods mentioned earlier in Functions Subsection 3.1 one-by-one. In that case, we get a low number of validated images. However, we can use a couple of methods together to find lines before the line cleaner combines the close lines. Testing one by one, we got the greatest number of validations using Smooth and Dent together. Out of 1778 images, the Smooth and Dent method together validated 1563 images. That makes above 85% success. If we try to add Sharp-v3 to the method group, the validation decreases to 915 images. However, if we compare both method groups, the Smooth + Dent method and Sharp-v3 + Smooth + Dent method validate the same 812 images.

On the other hand, the Sharp-v3 + Smooth + Dent method validated 103 images that Smooth + Dent could not validate by itself, while Smooth + Dent validated 751 images Sharp-v3 + Smooth + Dent could not validate. That shows us that using different combinations of methods gives different results for images. Even though the Smooth + Dent method has the most validation percentage among any combination, other combinations can validate the rest of the images Smooth + Dent cannot validate. Of course, as we mentioned several times, using Sharp methods, mostly Sharp-v1, hurts the reliability of the line detection. We will discuss the reliability of the lines later in this paper.

To validate the image, we use all the method combinations one by one, starting from including all the methods to using single ones. We start with more complex methods since if the complex method validates the image by finding only five lines, any sub-group of the combination can, at most, find five lines. If any sub-group finds fewer than five lines, we are

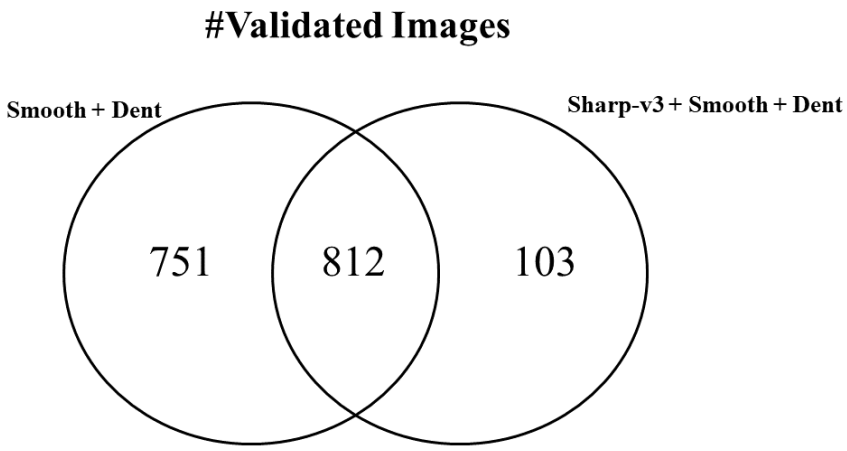


Figure 16: Venn diagram of the number of validated images by method.

not interested in that outcome. If any sub-group finds five lines, the only difference between the original group would be the middle point of the bands. Moreover, since we use a greater number of methods, we increase our reliability because more methods are agreed on those middle points. However, if our combined method finds more than five lines, we give its subgroups to find five lines. So, we have a better chance and reliability to validate the images if we use a more complex combination than simple ones.

Method	#Validated images	%Validated images
Sharp-v1	699	39.31%
Sharp-v2	479	26.94%
Sharp-v3	343	19.29%
Sharp-v1 + Sharp-v2 + Sharp-v3	479	26.94%
Smooth	43	2.42%
Dent	1542	86.73%
Smooth + Dent	1563	87.91%
Sharp-v3 + Smooth + Dent	915	51.46%
All together	479	26.94%
Combinations of methods	1747	98.25%

Table 1: #Validated Images out of 1778 images.

By using combinations of methods, we achieved more than 98% validity on 1778 images.

3.4 Reliability of the Lines

Finding the reliability of the band's middle points was the hardest thing we could not do. First, we tried to look at the symmetry of the bar values. If it is symmetric, then we would

say it has high reliability. The way to calculate the reliability was starting from the middle point of the line, taking the absolute difference of every pixel value with the same distance to the middle point in the n-neighborhood of the middle point and then taking the mean of the outputs and inserting the output into the Gaussian Error Function (2). That was the plan to calculate the reliability. Unfortunately, even though the middle point was correct on some images, the outcome gave us low reliability. So, we stopped trying to calculate the reliability there and focused on other issues.

$$\text{erf}z = \frac{2}{\sqrt{\pi}} \int_0^z e^{-t^2} dt \tag{2}$$

A way to improve reliability would be checking the lines found by each uncombined method. Suppose different methods have found one line. In that case, that can improve reliability. Suppose the line has been found by the Smooth method rather than Sharp-v1. In that case, we can increase the reliability since the Smooth method has the most reliable method. In contrast, Sharp-v1 has the least reliable one. However, these thoughts are still just in their initial stage, and these thoughts might be the next progress one can make to improve the code.

4 Results

Since we found the lines on the SP bar, we can start analyzing them. We start looking at the position of the lines on the SP bar. First, we normalize the line positions by dividing the bar length and multiplying by 100. That gives us what percentage is the line on the bar. As we can see in Table 2, the standard deviations of the lines are all around 7. For example, albumin's position is at 10% of the bar from the top and varies between 3% and 17%. This range is big to rely on the stability of the albumin's position. However, if we look at all the other lines, they all have a standard deviation near 7. The ratio of standard deviation to the mean of the bands means that lines do not stay at a specific position. However, lines may have a stable distance between them.

Line name	Mean	Std. deviation
albumin	10.22	6.97
α_1	27.86	7.91
α_2	47.46	8.00
β	62.89	7.26
γ	83.98	6.72

Table 2: Mean and standard deviation values of the validated images lines

So let us check the distance's standard deviation. After we check, we see the most stable ones are α_2 – β, albumin – α_1, and α_1 – α_2, with standard deviations of 10, 10, and 12, respectively. The distance value can vary more than twice the value compared to their mean values. So, we could not find any solid answer for the relationship between lines. We even checked the relation between any two distances, but the outcome was similar. The ratio of any two distances also has a large standard deviation relative to their mean.

Distance	Mean	Std. deviation
albumin - α_1	17.64	5.39
albumin - α_2	37.24	7.89
albumin - β	52.66	7.81
albumin - γ	73.76	8.93
α_1 - α_2	19.60	6.20
α_1 - β	35.02	7.17
α_1 - γ	56.12	9.13
α_2 - β	15.43	5.49
α_2 - γ	36.52	8.02
β - γ	21.09	6.59

Table 3: Mean and standard deviation values of the validated images line distances

Maybe there was a problem with our estimations, maybe we took the height of the bar wrong, or maybe there is no relation between the bands at all. If there would be a relation, we could take the ratios of the lines and figure out the bar height and the top of the bar easily. However, we could not find any ratio with our test, and there might be a ratio after all. The bands on the SP bar have a spectrum, whereas γ band has the biggest spectrum. Some results

show these spectrum borders between the bands as seen in Figure 17, which might help us locate the bar in the future. We need to check how electrophoresis works on blood samples for more information.

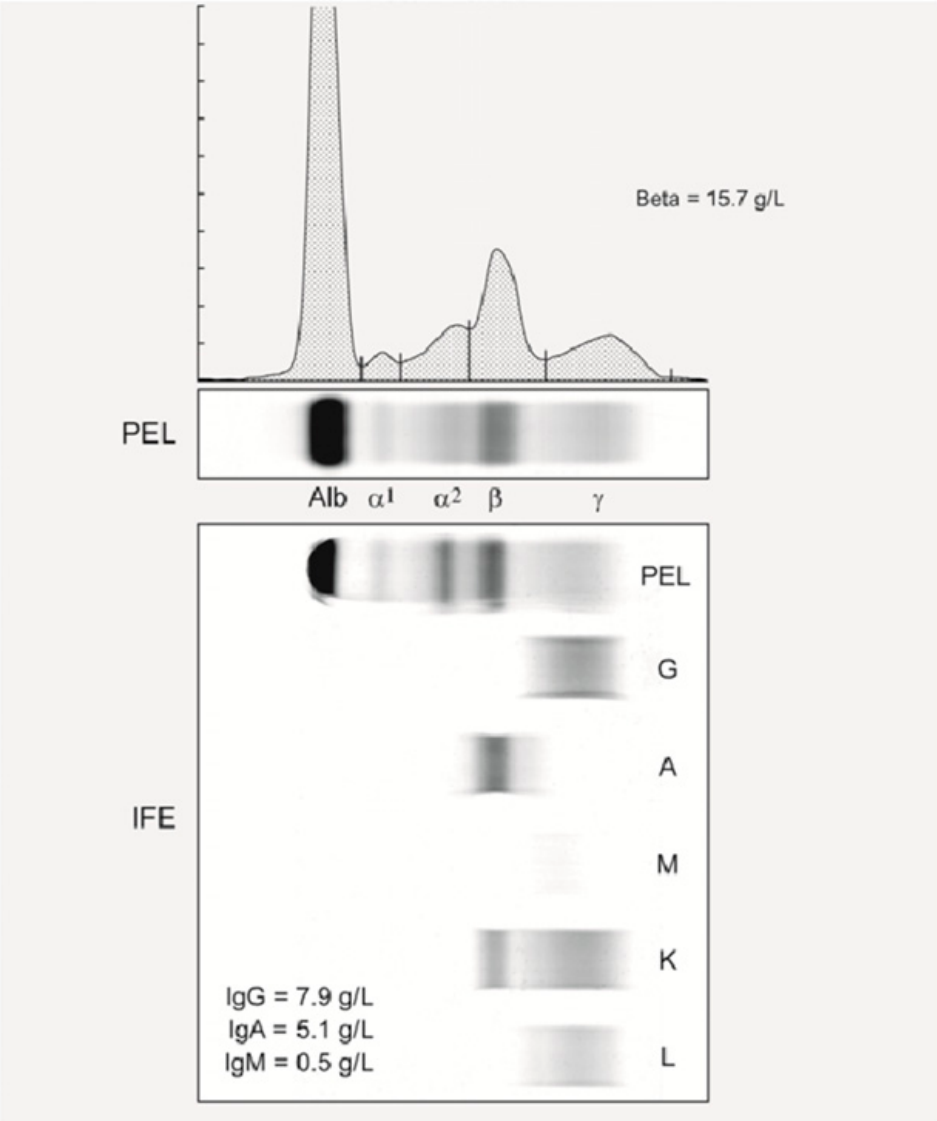


Figure 17: Image showing the boundaries between immunoglobulins on SP bar.Image is taken from *Chapter 8 : Clinical Tests for Monoclonal Proteins* by Leung [5].

5 Discussion

5.1 Further

Throughout our paper, we mentioned some improvements that can be made to what we have done until now. Here we summarize all for a compact view.

1. We need better or improved lined detection methods. Even though our combined methods can validate 98% of the data, we are still missing 2%. Understanding why we are missing the 2% of image data might help us to create a better method to detect lines and help the validated data to be more accurate in finding the correct middle points of the bands.

2. As we mentioned, we do not have a bar reliability value. Not having reliability for our lines makes our validation untrustworthy. Creating one would help us improve finding the bands' middle points on the SP bar. With that information, we could get close to finding a better mean and standard deviation of the band lines. As a result, we could find a relation between the bands, which would help to achieve better bar detection.

3. Ratio of the lines can be estimated better to improve finding bars on any size of an image by using the ratio. We used same-sized images to get the ratios, approximated with 1778 images without showing a reliability percentage. Finding better and more reliable ratios and using these ratios to detect bars would improve the use of data with different sizes.

5.2 What is Next?

Here, we began by using IFE images to identify the bars. After detecting each bar, we turned the bar images into structured data. From here, we need to process data to use for machine learning. For that, we need to create a classification model using LVQ machine learning methods and train the model. To compare the performance to Neural Network, we also need to create a Convolutional Neural Network (CNN) and feed it with image data. We train both models using the same initial data with a similar time frame to compare the performance. Furthermore, we can understand our models better than before with the comparison. One can even compare Neural Network, which uses structured data, and CNN, which uses initial image data, to understand which method is stronger for classifying image data where data have information to process.

6 Conclusion

In this paper, we have learned some new information about the SP bar on IFE data. Even though we could not find any stable position of the bands on the SP bar, we answered that there is no strict relation between the bands. Also, we opened a question on how to use the approximated band position of the SP bands to estimate the bar position better. On the other hand, we have successfully extracted bar graphs in images into structured data where we can use the data for machine learning methods other than Neural Networks, such as LVQ. With that, we started our first step to use the data in a machine learning method other than CNN. Overall, these findings offer valuable insights into the use of machine learning in processing IFE data and suggest future avenues for research in this area.

Declaration

I hereby declare that I have prepared the present work independently and only using the literature and aids indicated. Passages taken literally or analogously from sources are marked as such.

Mittweida, 03 April 2023

Mert Saruhan

References

[1] MedlinePlus [Internet]. “Immunofixation IFE Blood Test”. MedlinePlus, 15 Nov. 2021, www.medlineplus.gov/lab-tests/immunofixation-ife-blood-test/.

[2] Testing [Internet]. “Protein Electrophoresis, Immunofixation Electrophoresis”. Testing, 09 Nov. 2021, www.testing.com/tests/protein-electrophoresis-immunofixation-electrophoresis.

[3] Tuve, Sebastian, et al. “Deletion 13q14 in Plasma Cells of a Patient with Lupus Erythematosus and Autoimmune Hepatitis”. Journal of Bone Marrow Res, vol. 2, no. 2, 2014, pp. 1,2, https://doi.org/10.4172/2329-8820.1000147.

[4] Wahed, Amer, and Amitava Dasgupta. “Hematology and Coagulation: A Comprehensive Review for Board Preparation, Certification and Clinical Practice”. 1 st ed., Elsevier, 2015, https://doi.org/10.1016/B978-0-12-800241-4.00007-3.

[5] Leung, Nelson R. “Chapter 8 : Clinical Tests for Monoclonal Proteins” (2016)

[6] Bhutta, Riaz Ahmad, et al. ”Monoclonal Immunoglobulin (Ig), Monoclonal antibody, Immunofixation Electrophoresis (IFE)”, Labpedia, 25 Jan. 2020, https://labpedia.net/monoclonal-immunoglobulin-ig-monoclonal-antibody-immunofixation-electrophoresis-ife/

[7] Eclinpath [Internet]. “Protein Electrophoresis”. Eclinpath, www.eclinpath.com/chemistry/proteins/electrophoretic-patterns/.