Annals of Biomedical Research
An open access journal
eSciRes
ISSN 2817-2094
Research Article ABR-5-126

A Deep-Learning Error Detection System in Radiation Therapy

Kump PM[1*], Xia J[2], Yaddanapudi S[3] and Bai E[4]
[1]Department of Electrical and Computer Engineering, College of Engineering, Kansas State University, Manhattan, KS, USA
[2]Department of Radiation Oncology, Mount Sinai Hospital, New York City, NY, USA
[3]Department of Radiation Oncology, College of Medicine, University of Iowa, Iowa City, IA, USA
[4]Department of Electrical and Computer Engineering, College of Engineering, University of Iowa, Iowa City, IA, USA

Abstract

Delivering radiation therapy based on erroneous or corrupted treatment plan data has previously and unfortunately resulted in severe, sometimes grave patient harm. Aiming to prevent such harm and improve safety in radiation therapy treatment, this work introduces a novel, yet intuitive algorithm for strategically structuring the complex and unstructured data typical of modern treatment plans so their treatment sites may automatically be verified with deep-learning architectures. The proposed algorithm utilizes geometric and dose plan parameters to represent each plan's data as a heat map to feed a deep-learning classifier that will predict the plan's treatment site. Once it is returned by the classifier, a plan's predicted site can be compared to its documented intended site, and a warning raised should the two differ.

Using real head-neck, breast, and prostate treatment plan data retrieved at two hospitals in the United States, the algorithm is evaluated by observing the accuracy of convolutional neural networks (ConvNets) in correctly classifying the structured heat map data. Many well-known ConvNet architectures are tested, and ResNet-18 performs the best with a testing accuracy of 97.8% and 0.979 F-1 score. Clearly, the heat maps generated by the proposed algorithm, despite using only a few of the many available plan parameters, retain enough information for correct treatment site classification. The simple construction and ease of interpretation make the heat maps an attractive choice for classification and error detection.

Keywords: Radiation therapy error detection; Treatment plan data structure; Deep learning; Transfer learning

Abbreviations

API: Application programming interface; ConvNet: Convolutional neural network; CT: Computerized tomography; LINAC: Linear accelerator; ML: Machine learning; MLC: Multi-leaf collimator; RGB: Red-green-blue ReLU: Rectified linear unit; ResNet: Residual network; SQL: Structured query language; VGG: Visual geometry group

Introduction

The process of planning and delivering radiation therapy treatment has seen many recent technological advancements which provide cancer patients with unprecedented new cures but also introduce new pathways towards causing harm. The increasing complexity of evolving technologies opens the door to a whole host of potential errors in the treatment delivery pipeline [1], arising from faulty programming, software flaws, and inadequately trained staff [2], for example. Both the enthusiasm of hospitals and clinics to usher in cutting-edge, but unproven technologies [3] and the accelerating trend of radiation as a cancer treatment exacerbate the likelihood of such errors and the severe or fatal results that potentially follow.

A breast cancer treatment patient died in 2007 from receiving a lethal dose of radiation due to a programming error compromising the linear accelerator (LINAC)—a machine that generates beams of high-energy radiation. In this case, the error instructed the LINAC to deliver more than three times the prescribed dose during each radiotherapy session, which, shockingly, went undetected for 30 sessions [3]. In a separate incident from the same year, a second treatment patient died from similar circumstances in which a computer error directed the LINAC to behave erroneously [4]. Not once, but on three separate occasions the errant LINAC blasted with radiation the patient's brain stem instead of the tongue, which was the intended treatment site. Other examples include 16 patients in Panama who were severely overexposed, resulting in eight treatment-related deaths [5]. These incidents exemplify the need for the methods presented in our study, which aims to automatically detect a mismatch between the intended and actual treatment sites (e.g., head-neck, breast, prostate). Upon detecting a mismatch, the system can alert medical personnel and promote manual intervention, potentially preventing future incidents like those described above.

The complex radiation therapy process consists of many steps from initial patient consultation to final radiation treatment. If the patient is to receive treatment therapy as determined during consultation with the oncologist, a customized treatment plan is developed for the patient, originating from a simulation process that includes a computerized tomography (CT) scan to image the tumor. After the oncologist outlines the tumor in the image and at-risk organs to be avoided, a computerized system designs the finalized treatment plan, which is necessarily intricate and detailed. The treatment plan, which varies by patient, clinic, oncologist and LINAC vendor, is then reviewed and approved by the oncologist and subsequently checked by medical physicists for safety and quality. It is here in the treatment process flow that we propose to insert our automatic error-checker, before the last step in the process: treating the patient with radiation from the LINAC.

Successful development of such an automatic error-checking system depends strongly on the quality and quantity of the data used to train it. Conveniently, healthcare systems collect and store vast amounts of data [6], and standard practice in radiation oncology specifically is to store treatment plan data in an oncology database [7]. We leverage this fact and choose to formulate our detection scheme in the context of machine learning (ML)—specifically deep learning, which has previously been deployed with great success in the medical field [6,8-9]. However, treatment plan data unfortunately lack structure and certainly may not feed deep-learning algorithms, or any other intelligent algorithm, in their raw form. Moreover, the parameters describing treatment plans are different by clinic, and development of an error-checking system requires a unified treatment plan data structure.

Therefore, the major contribution of our work is in the structuring of radiation treatment plan data so they may easily train previously established ML algorithms. The work is the first step towards the goal of an automated software suite or dashboard for plan diagnostics and error-checking. We identify only the relevant information stored in oncology databases since much of that information, it turns out, is irrelevant towards that goal. To maximize the potential of this work, we unify our data structure across all clinics and LINACs, which is necessary since data in oncology databases are non-uniform and represented in vendor-specific formats that depend on the specific LINAC used to deliver treatment [10]. Furthermore, our proposed data structure may be interpreted as an image, useful for feeding powerful, off-the-shelf computer vision algorithms.

Background

Understanding our data structure requires a brief understanding of the LINAC's construction. Shaping the beam radiated by the LINAC is its multi-leaf collimator (MLC) consisting of two opposing banks of mechanical tungsten leaves situated in a rectangular area defined by two opposing pairs of horizontal and vertical jaws, see Figure 1. Forming an aperture through which the radiation passes, the leaf and jaw positions are adjustable, with positions automatically determined by the computerized treatment planning system. The MLC is allowed to rotate in the plane containing the leaves and jaws, and its angle of rotation is also determined by the planning system. As the gantry, with its attached MLC, rotates around the patient to deliver the radiation treatment, the leaf and jaw positions, as well as the MLC angle, are allowed to change—a fact that contributes to the plan's complexity. The strength of the radiation beam (i.e., the "dose") may also change throughout treatment delivery and is also determined by the planning system. Parameters describing the dose and the exact configuration of the MLC at every point in time are just some of the data that represent a single treatment plan.

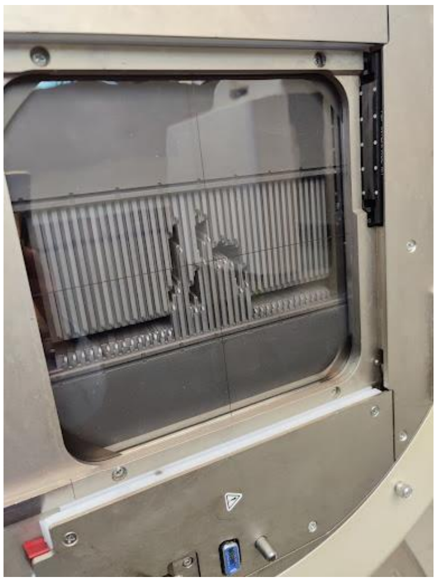


Figure 1: Multi-leaf collimator (Varian TrueBeam LINAC).

The purpose of the MLC—with its leaves and jaws—is to prevent damage to the healthy tissue that surrounds the cancerous tumor by shaping the radiated beam appropriately. Intuition suggests that treatment sites like prostate, head-neck, etc., are each uniquely shaped, and the geometric configuration of the MLC is indicative of this site. We therefore structure each treatment plan as a heat map describing the spatial distribution of radiation dose as it leaves the LINAC through its aperture. Our heat maps are generated from plans' jaws, leaf and dose parameters, wholly available from oncology databases. We evaluate our heat map structure by training and testing many deep-learning architectures' ability to predict a plan's treatment site given its heat map, and we find that ResNet-18, with an accuracy of 97.8% and F-1 score of 0.979, performs very well.

Instead of analyzing complex treatment plans that are typical in the field of modern oncology, previous work in this area analyzes simple four-field box plans in which the radiated beam is confined to a simple aperture [11,12]. The authors in [11] apply K-means clustering to detect treatment plan errors in four-field box plans, and only in prostate cancer at a single institute. More recently, the authors in [12] verify the clinical acceptability of four-field box plans by applying deep-learning models that are trained on simple apertures and digitally reconstructed radiographs. These methods and others [13] stop short of classifying treatment site and instead focus only on identifying anomalous treatment plans. Our work, on the other hand, considers the intricate and comprehensive treatment plans more relevant to modern technologies. Our best results, prior to this study, employed a primitive version of our heat map idea to feed classical ML algorithms [14,15] instead of deep-learning ones. The results were an improvement over our original method [16] that did not use heat maps at all.

The rest of this paper is organized as follows: Section 3 describes the data collection process and how we construct heat maps, including associated computational effort. We also report our procedure for augmenting the data set. Section 4 provides a description of the deep-learning models that we consider, and the classification results they yield. We also provide an analysis of our results in Section 4, which includes insight into what the architectures learn and why certain plans are misclassified. We summarize our results, acknowledge the study's limitations, and provide paths for future research in Section 5.

3. Materials and Methods

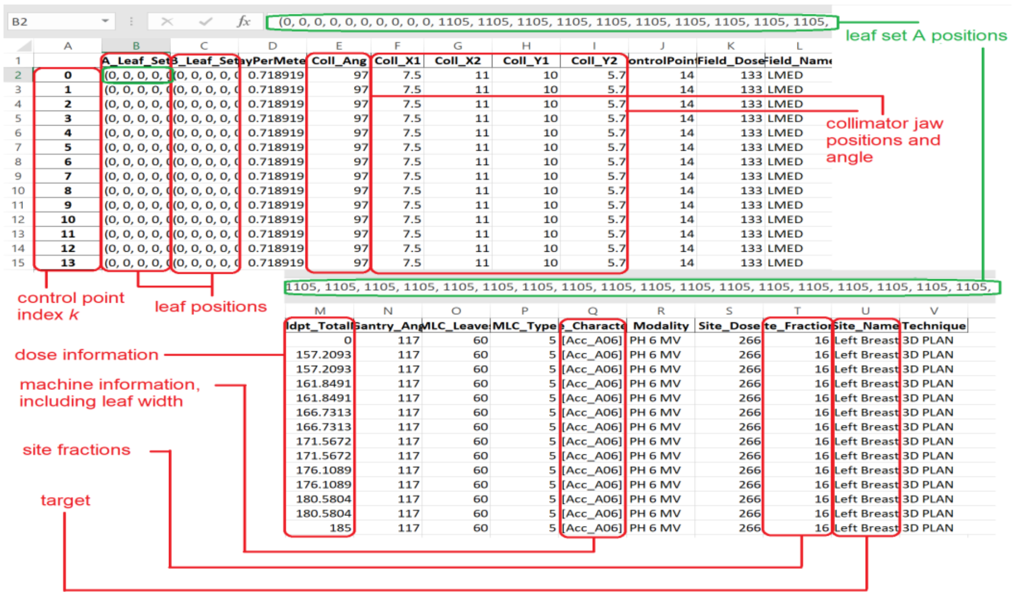
3.1 Data Collection

By means of custom Structured Query Language (SQL) queries, treatment plan data from two clinics in the United States are retrieved from the MOSAIQ (Elekta AB, Stockholm, Sweden) oncology information system. Each treatment plan is converted to a Python dictionary for analysis. In practice, treatment plans are originally generated from multiple vendor systems, with those from one clinic by Eclipse treatment planning Varian Medical Systems and the other clinic, the Pinnacle (Philips Healthcare) system. Together, the two clinics account for all 697 treatment plans analyzed in the study, which, for the sake of moderating the study's scope, are limited to head-neck, breast, and prostate treatment sites. Table 1 shows the count of each treatment site considered in the study before and after data augmentation, which is discussed in Section 3.4. The data set is biased in breast cancer plans, a fact motivating our use of class-weighted F-1 scores for assessing our method in Section 4.2.3. Though the data are limited to only these three treatment sites, the method that follows can be generalized to more treatment sites given appropriate treatment plan data.

	Head-neck	Breast	Prostate	Total
Initially	186	305	206	697
After augmentation	2046	3355	2266	7667

Table 1: Number of plans for each treatment site in the data set before and after data augmentation.

The raw treatment plan data from the oncology databases are complex and unstructured, with treatment plans characterized by thousands of parameters. The exact positions and orientations of the gantry, collimator, and leaves, for example, necessitate many parameters to describe precisely, especially considering they may change throughout the delivery of radiation in a single plan. Illustrating a treatment plan as an Excel file, Figure 2 demonstrates that a single treatment plan is effectively a listing of these parameters and many other parameters that are relevant for accurate delivery of treatment. However, no universal set of parameters exists. This fact contributes to the challenge of representing the data in a structured manner. In our study, for example, two different LINAC manufacturers each utilizes its own parameters and conventions. Table 2 shows machines can employ either NL =60 leaves or NL =80 in each of its two MLC leaf banks A and B. Leaf widths can also vary by machine. Additionally, some machines have two pairs of opposing collimator jaws, whose positions are described by the parameters Coll_X1, Coll_X2, Coll_Y1, and Coll_Y2; other machines lack the horizontal pair and the Coll_X1 and Coll_X2 data.



(0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1105, 1105, 1105, 1105, 1105, 1105, 1105, 1105, 1105, 1105,

	A	B	C	D	E	F	G	H	I	J	K	L
1		A_Leaf_Set	3_Leaf_Set	ayPerMete	Coll_Ang	Coll_X1	Coll_X2	Coll_Y1	Coll_Y2	ontrolPoin	Field_Dose	ield_Name
2	0	(0, 0, 0, 0, (	(0, 0, 0, 0, (	0.718919	97	7.5	11	10	5.7	14	133	LMED
3	1	(0, 0, 0, 0, (	(0, 0, 0, 0, (	0.718919	97	7.5	11	10	5.7	14	133	LMED
4	2	(0, 0, 0, 0, (	(0, 0, 0, 0, (	0.718919	97	7.5	11	10	5.7	14	133	LMED
5	3	(0, 0, 0, 0, (	(0, 0, 0, 0, (	0.718919	97	7.5	11	10	5.7	14	133	LMED
6	4	(0, 0, 0, 0, (	(0, 0, 0, 0, (	0.718919	97	7.5	11	10	5.7	14	133	LMED
7	5	(0, 0, 0, 0, (	(0, 0, 0, 0, (	0.718919	97	7.5	11	10	5.7	14	133	LMED
8	6	(0, 0, 0, 0, (	(0, 0, 0, 0, (	0.718919	97	7.5	11	10	5.7	14	133	LMED
9	7	(0, 0, 0, 0, (	(0, 0, 0, 0, (	0.718919	97	7.5	11	10	5.7	14	133	LMED
10	8	(0, 0, 0, 0, (	(0, 0, 0, 0, (	0.718919	97	7.5	11	10	5.7	14	133	LMED
11	9	(0, 0, 0, 0, (	(0, 0, 0, 0, (	0.718919	97	7.5	11	10	5.7	14	133	LMED
12	10	(0, 0, 0, 0, (	(0, 0, 0, 0, (	0.718919	97	7.5	11	10	5.7	14	133	LMED
13	11	(0, 0, 0, 0, (	(0, 0, 0, 0, (	0.718919	97	7.5	11	10	5.7	14	133	LMED
14	12	(0, 0, 0, 0, (	(0, 0, 0, 0, (	0.718919	97	7.5	11	10	5.7	14	133	LMED
15	13	(0, 0, 0, 0, (	(0, 0, 0, 0, (	0.718919	97	7.5	11	10	5.7	14	133	LMED

1105, 1105, 1105, 1105, 1105, 1105, 1105, 1105, 1105, 1105, 1105, 1105, 1105, 1105, 1105, 1105, 1105, 1105,

M	N	O	P	Q	R	S	T	U	V
ldpt_Total	Gantry_Ang	MLC_Leave	MLC_Type	_Characte	Modality	Site_Dose	te_Fraction	Site_Name	Technique
0	117	60	5	[Acc_A06]	PH 6 MV	266	16	Left Breast	3D PLAN
157.2093	117	60	5	[Acc_A06]	PH 6 MV	266	16	Left Breast	3D PLAN
157.2093	117	60	5	[Acc_A06]	PH 6 MV	266	16	Left Breast	3D PLAN
161.8491	117	60	5	[Acc_A06]	PH 6 MV	266	16	Left Breast	3D PLAN
161.8491	117	60	5	[Acc_A06]	PH 6 MV	266	16	Left Breast	3D PLAN
166.7313	117	60	5	[Acc_A06]	PH 6 MV	266	16	Left Breast	3D PLAN
166.7313	117	60	5	[Acc_A06]	PH 6 MV	266	16	Left Breast	3D PLAN
171.5672	117	60	5	[Acc_A06]	PH 6 MV	266	16	Left Breast	3D PLAN
171.5672	117	60	5	[Acc_A06]	PH 6 MV	266	16	Left Breast	3D PLAN
176.1089	117	60	5	[Acc_A06]	PH 6 MV	266	16	Left Breast	3D PLAN
176.1089	117	60	5	[Acc_A06]	PH 6 MV	266	16	Left Breast	3D PLAN
180.5804	117	60	5	[Acc_A06]	PH 6 MV	266	16	Left Breast	3D PLAN
180.5804	117	60	5	[Acc_A06]	PH 6 MV	266	16	Left Breast	3D PLAN
185	117	60	5	[Acc_A06]	PH 6 MV	266	16	Left Breast	3D PLAN

Figure 2: Treatment plan viewed as an Excel spreadsheet. A single plan may populate more than one sheet.

In the following section, we overcome these difficulties by structuring treatment plan data as a radiation heat map as in Figure 3. The heat maps' length and width are uniform across all treatment plans, independent of the machine delivering the radiation. After some trial and error experimenting with other methods, we devise the idea of a heat map from our conjecture that each of the three treatment sites is different in shape and radiation dose, consequently demanding a unique spatial distribution of dose to the patient. Our heat maps are two-dimensional and use only those data listed in Table 2 despite the many parameters comprising the treatment plans. Of course, the heat maps do not utilize Site Name, which is ultimately to be predicted by the classifier. Given their image-like structure, heat maps enjoy the added benefit of easily feeding deep-learning architectures for classification.

Parameter identifier	Description	Unit	Type
A_Leaf_Set	A vector of length 60 or 80 leaf positions in Bank A	Tenths of mm	Vector of ints
B_Leaf_Set	A vector of length 60 or 80 leaf positions in Bank B	Tenths of mm	Vector of ints
Coll_Ang	Angle of collimator rotation	Degrees	Int
Coll_X1	Left position from center of collimator jaws	cm	Int
Coll_X2	Right position from center of collimator jaws	cm	Int
Coll_Y1	Bottom position from center of collimator jaws	cm	Int
Coll_Y2	Top position from center of collimator jaws	cm	Int
Dose	Amount of radiation delivered from the LINAC per fraction	centi-Gray	Float
Leaf_Widths	A vector of length 60 or 80 leaf widths	mm	Vector of floats
Site_Fractions	The number of times the treatment plan repeats	No unit	Int
Site_Name	Treatment site	N/a	String

Table 2: Treatment plan data retrieved in raw form from oncology databases. Only shown are data used in this study.

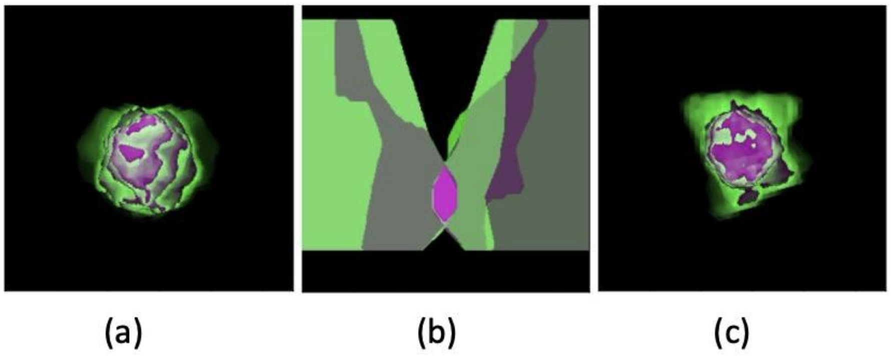


Figure 3: One typical heat map of each treatment site: (a) head-neck, (b) breast, and (c) prostate. Most breast heat maps appear rectangular, as shown, but some look like head-neck and prostate.

3.2 Heat Map Construction

Treatment delivered by a LINAC occurs in discrete steps as its gantry rotates around the patient, which may span several sessions occurring over many days. During this process, many of the parameters in Table 2 have the potential to change at each step k, including dose, jaw and leaf positions, and collimator angle. A treatment plan is comprised of these N steps, which can be as few as $N = 1$or as many as $N = 100$ or more. The idea is to pixelate the kth aperture, $k = 1, \ldots, N$, and accumulate the dose radiating through all N apertures.

At a high level, heat maps are calculated according to the following process, which is also illustrated in Figure 4. Each item is later explained in detail.

1. At each k, calculate the kth aperture and rotate it by the k^{th} collimator angle.
2. Accumulate the dose through each of N rotated apertures. The result is a single- channel image.
3. Crop, down-sample, and filter the image.
4. Create red, green, and blue (RGB) channels utilizing site fractions and channel replication.

3.2.1. Apertures: An aperture refers to the two-dimensional surface through which the radiation beam is allowed to pass unobstructed by the collimator jaws and leaves. Its complex shape is completely specified by leaf widths, leaf positions, and jaw positions. For convenience, we ignore for now that each plan has N apertures, one at each step k, and instead focus on constructing a single aperture.

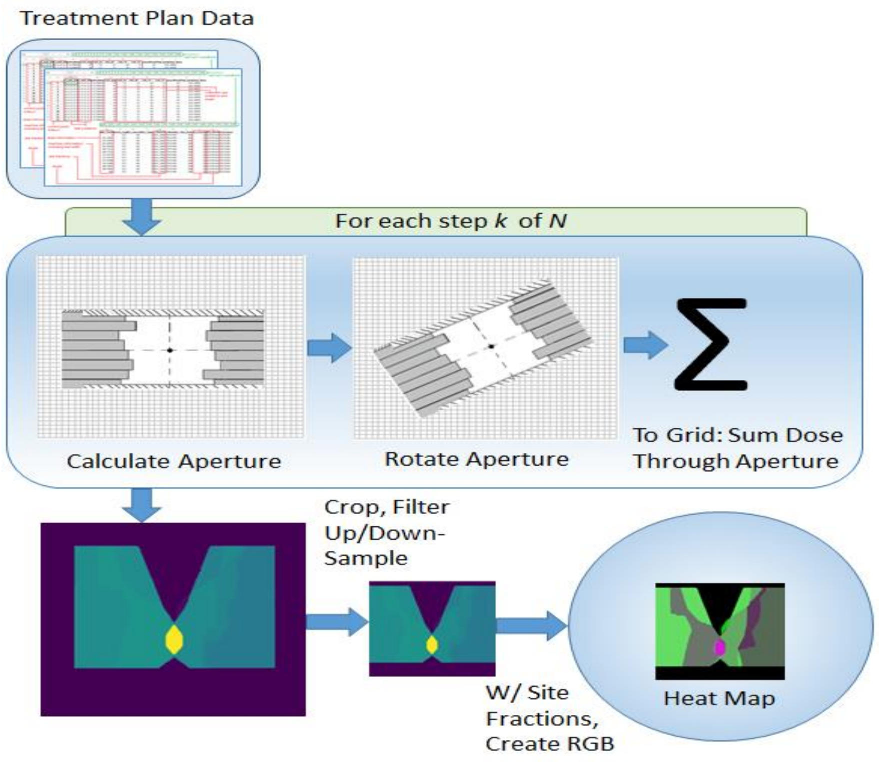

Treatment Plan Data
For each step k of N
Calculate Aperture
Rotate Aperture
To Grid: Sum Dose Through Aperture
Crop, Filter Up/Down-Sample
W/ Site Fractions, Create RGB
Heat Map

Figure 4: Process flow for creating proposed treatment plan heat maps.

Later, we will amend the result to include all N apertures. The key to pixelating an aperture is identifying the smallest unit of distance in the data set, which is Δt = 0.1 mm in our data set. Every length in the data set can be expressed as an integer multiple of Δt. Then, we establish a pixel grid G with square pixels of length Δt to represent the aperture exactly.

Consider a collimator oriented like in Figure 5(a), with leaf widths aligned with the vertical, and leaf positions, the horizontal. Leaf positions are referenced in the data set by their distance away from the y axis, which falls at the midpoint of the collimator. In units of Δt, any leaf may take on one of 2,000 equally spaced positions to the left of the y axis, 2,000 to the right of the y axis, or the y axis itself. Imagining a vertical line at each of these positions creates 2,000 columns to the left of the y axis and 2,000 to the right, for a total of 4,000 columns, each Δt wide. We choose to index the columns as $l = -2000, \ldots, 1999$ to center (approximately) the $l = 0$ column.

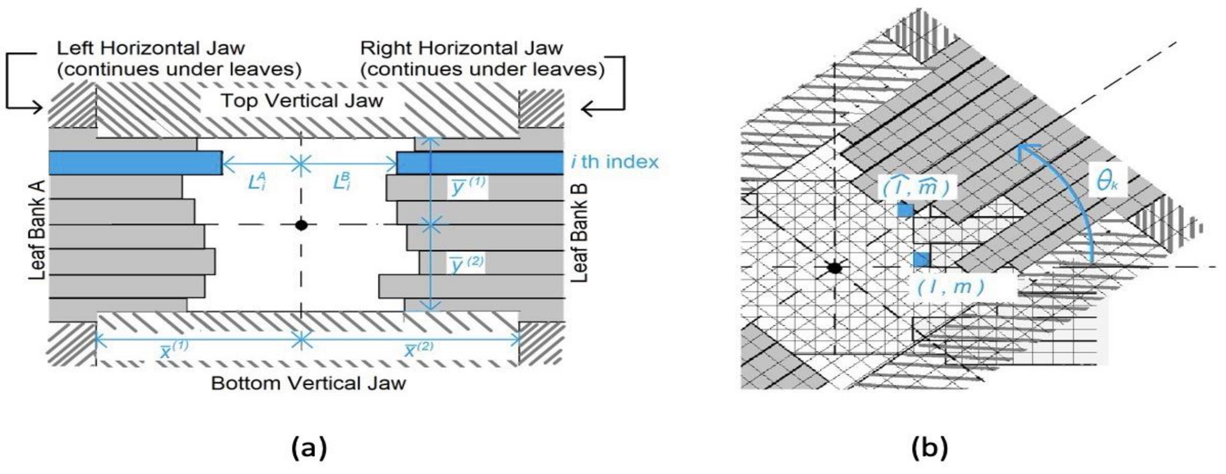


Figure 5: (a) Example aperture formed by the leaves in Banks A and B and the rectangular collimator jaws. (b) Pixelated aperture shown with collimator rotation of angle θ_k. A pixel at location (l, m) before rotation is located at $(\hat{l}, \hat{m})$ after rotation.

The row indexing, on the other hand, may be understood by considering the leaf widths. Let w_i represent the width, in units of pixels, of the ith leaf, $i = 1, \ldots, N_L$. The total width of any leaf bank $\Sigma_i\, w_i$ in our data set is either 4000Δ*t* or 2200Δ*t*, depending on the machine. We therefore establish 4,000 rows to accommodate the larger of the two collimators. Thus, the size of the grid, which must remain constant no matter the machine, is 4,000 by 4,000. We choose to index the rows as $m = -2000,\ldots,1999$ to center (approximately) the $m = 0$ row.

Since leaf widths vary by leaf, it is useful to define the indicator sequence

$$R_m = \overbrace{\underbrace{1}_{m=-2000}\quad,\ldots,1}^{w_1},\overbrace{2,\ldots,2}^{w_2},\overbrace{3,\ldots,3}^{w_3},\ldots,\overbrace{N_L,\ldots,N_L}^{w_{N_L}}$$

This sequence R_m maps the grid row index m to the leaf index i. For example, if Leaf 1, the bottom leaf, has a width of $w_1 = 25$ pixels, then $R_m = 1$ for $m = -2000$ to $m = -1976$. The value of R_m indicates Leaf 1 is responsible for shaping the aperture at these mth row locations. The length of R_m is only 2200, not 4000, for machines with banks of total width 2200Δ*t*. For these cases, we up-sample R_m by a factor of 4000/2200 = 20/11 to achieve uniform length across all machines. Alternatively, the longer R_m may be down-sampled by a factor of 11/20 so that R_m has a uniform length of 2200. In that case, the grid contains 2200 rows, and index $m = -1100, \ldots, 1099$.

Let $L^A{}_i$ and $L^B{}_i$ be the length in pixels measured from the y axis to the edge of the ith leaf in Banks A and B, respectively. Also let $\bar{x}^{(1)}$ and $\bar{x}^{(2)}$ be the length in pixels measured from the y axis to the edge of the left and right horizontal collimator jaws, respectively. As in Figure. 5 (left), $L^A{}_i > 0$ in the data set if the leaf extends to the left of the y axis, and $L^B{}_i > 0$, the right. Similarly, $\bar{x}^{(1)} > 0$ if the jaw is located to the left of the y axis, and $\bar{x}^{(2)} > 0$, the right. Data from other oncology databases may adhere to different conventions, but data are easily programmatically adjusted to conform.

The ith leaf in Bank A shapes the aperture only if the left jaw position $\bar{x}^{(1)}$ is recessed further than the leaf; otherwise, the jaw shapes the aperture. So the quantity $\max(-L^A{}_i\ ,\ -\bar{x}^{(1)})$ provides the smallest column index l defining the left side of the aperture at the *i*th leaf. Similarly, the quantity $\min(L^B{}_i\ ,\ \bar{x}^{(2)}) - 1$ provides the largest column index l defining the right side of the aperture at the *i*th leaf. It is useful then to define the set

$$C = \{\max(-L^A{}_i\,, -\bar{x}^{(1)}), \ldots, \min(L^B{}_i\,, \bar{x}^{(2)}) - 1\},$$

which describes the column indices of pixels inside the aperture at the ith leaf.

The vertical jaws play a similar role as the horizontal jaws in forming the aperture. The top of the aperture is given by the position in pixels of the top vertical collimator jaw $\bar{y}^{(1)}$. Above this position, the jaw masks the leaves, and their positions do not affect the aperture. Similarly, the bottom of the aperture is given by the position in pixels of the bottom vertical collimator jaw $\bar{y}^{(2)}$. Like in Figure 5(a), the sign convention is $\bar{y}^{(1)} > 0$ if the top jaw is positioned above the horizontal axis, and $\bar{y}^{(2)} > 0$, below the axis.

The pixel grid $G(l, m)$ is to describe the dose d through the aperture. It may now be defined as follows:

Define grid $G(l\ ,m) = 0$ for $l, m = -2000, \ldots, 1999$.

For each row index $m = -\bar{y}^{(2)}, \ldots, \bar{y}^{(1)} - 1$:

Get the leaf index i given by $R_m$and then leaf positions $L^A{}_i$ and $L^B{}_i$. **If** $\max(-L_i^A, -\bar{x}^{(1)}) = \min(L_i^B, \bar{x}^{(2)})$, the aperture is closed. Do nothing. **Otherwise, for** each l in set C: Set $G(l, m) = d$.

Thus, pixels within the aperture contribute an amount d to D, and those outside contribute nothing.

Until this point, we have ignored that dose, leaf and jaw positions are all functions of step k as the gantry rotates around the patient. We did so for ease of notation but may easily replace the parameters in the previous algorithm with $d_k, L_{i,k}^A, L_{i,k}^B, \bar{x}_k^{(1)}, \bar{x}_k^{(2)}, \bar{y}_k^{(1)}$, and $\bar{y}_k^{(2)}$. The resulting grid is denoted as $G_k(l, m)$.

Finally, for machines without horizontal jaws, we artificially set values of $\bar{x}_k^{(1)}$ and $\bar{x}_k^{(2)}$ so they do not affect the aperture. That is, we choose $\bar{x}_k^{(1)}$and $\bar{x}_k^{(2)}$ so

$$\max(-\bar{x}_k^{(1)}, -L_{i,k}^A) = -L_{i,k}^A,$$

$$\min(\bar{x}_k^{(2)}, L_{i,k}^B) = L_{i,k}^B, \quad \forall i, k.$$

These conditions ensure the horizontal jaw positions are recessed more than any single leaf in either bank and do not contribute to the formation of $G_k(l, m)$.

3.2.2. Rotation: Before the G_k's can be summed together to obtain a heat map, the collimator's rotation angle ϑ_k at control point k should be considered. Otherwise, such a sum would inaccurately describe the dose distribution leaving the LINAC as illustrated in Figure 5(b). The mapping between original pixel coordinates (*l, m)* and rotated coordinates $(\hat{l},\ \hat{m})$is based on a rotation matrix so that

$$\hat{l} = \lfloor \mathrm{l} \cdot \cos\theta_k - \mathrm{m} \cdot \sin\theta_k \rfloor,$$

$$\hat{m} = \lfloor \mathrm{l} \cdot \sin\theta_k + \mathrm{m} \cdot \cos\theta_k \rceil.$$

The notation $\lfloor\cdot\rceil$ represents rounding to the nearest integer, which is necessary for discrete pixel locations. The transformation assumes a rotation about the origin. A treatment plan's heat map H is constructed by rotating each G_k and accumulating over each of the N control points comprising the plan:

$$H(\hat{l},\hat{m}) = \sum_{k=1}^{N} G_k(l,m).$$

3.2.3. Down-Sampling, Filtering: The resolution of each heat map is too high for our modest computer system (whose specifications are provided in Section 3.3). Accordingly, we subsequently crop and down-sample each heat map to reduce the resolution. We choose to symmetrically crop each heat map 15% from the top, bottom, left and right edges since the coordinate system was chosen for a centered aperture. As illustrated in Figure 3, cropping the heat maps in this way retains much of the information.

The 2,800-by-2,800 resolution of the cropped heat maps is still too large, it turns out, for our computer system. Preferring not to crop any more than 15% from each edge, we opt to down-sample each cropped map by a factor of 10 to further decrease the resolution to 280-by-280. Down-sampling in this way allows us to process the data but also aliases the maps with noise. The top row of Figure 6 indicates increased high-frequency noise in the down-sampled image. Corresponding to this increase in high frequencies, the bottom pane Figure 6(a) shows a close-up view of the salt-and-pepper noise resulting from the down-sampling process. The noise is easily removed with a simple 5-by-5 median filter, as indicated by the bottom-right pane. The median filter is applied to each heat map after cropping.

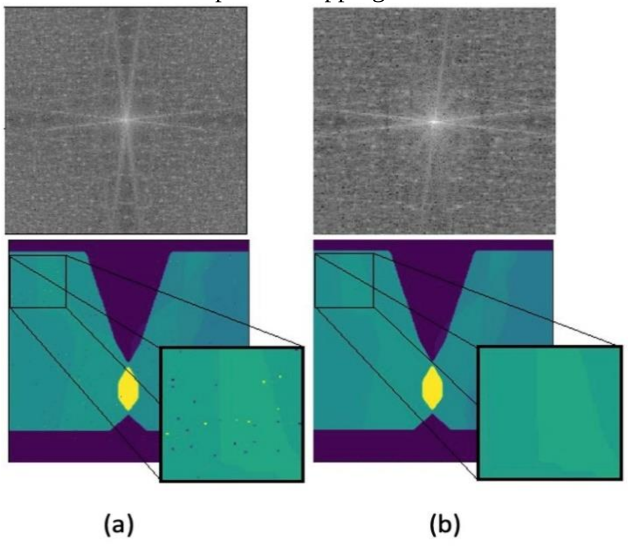


Figure 6: Analysis of one channel of a single selected breast plan to illustrate aliasing and filtering. (a, top:) Fourier transform of the plan's heat map after cropping and before down-sampling. (b, top:) Fourier transform of the plan's heat map after down- sampling, showing increased intensity at high frequencies in the center. (a, bottom:) The plan's heat map with noise from aliasing. (b, bottom:) The plan's heat map after smoothing with a median filter.

In the subsequent text, H refers to this down-sampled and filtered final iteration of heat maps for ease of notation.

3.2.4. Site Fractions, Channel Replication: In practice, a plan's total dose is typically administered in equal parts, or fractions, during separate treatment sessions. Radiation is repeated s number of sessions as determined by the oncologist and documented in the plan's data, see the parameter Site_Fractions in Table 2. The heat map H therefore represents a plan's fractional dose of radiation because the dose parameter is the dose per fraction. Thus, the product sH provides the plan's total dose.

It is important to include s into the data structure to distinguish plans having similar fractional dose but different total dose, and vice-versa. Since multiplying by the scalar constant s does not change the dimensions of H, we are able to incorporate s by forming three channels as in RGB color images. We form each heat map into a volume of size (3, 280, 280) in which the first channel is H, the second is sH, and the third is again H. In this way, the image-like structure is preserved, and the value of s may be discerned by comparing the middle channel with either of the other two channels. The choice of using either H or sH for the third channel is arbitrary and does not incorporate any new information into the structure.

3.3 Computation Effort: Heat Maps

Using Spyder running Python 3.7, we construct heat maps on a modest Microsoft Surface Book with an Intel Core i7 processor at 2.60GHz, 16 GB of RAM, and NVIDIA GeForce Graphics Processing Unit (GPU). For memory efficiency, upon computing each plan's final heat map (i.e. cropping, down-sampling, and filtering), we programmatically delete all intermediate variables before moving on to the next plan's heat map. The 697 heat maps are computed in a total of ~9 hours, or about (9/697) × 3600 = 46.5 seconds per plan, after which the collection of heat maps is saved as a Python pickle file to be loaded into a more powerful machine for algorithm training and testing (specifications described in Section 4.3). Computing and storing the heat maps pushes the machine to its limits, at times demanding 99% of the machine's memory.

3.4 Data Augmentation

Data augmentation is the process of expanding the data set by engineering new images that are distorted versions of the original images. Data augmentation has been shown to increase prediction performance, especially in medical imaging [17], when the original data set is insufficient in size and the engineered images can provide useful information to the classifier. Accordingly, we leverage this technique for two reasons: (1) Our data set of 697 treatment plans is most likely insufficient to train accurate deep-learning architectures, and (2) small variations in the shapes of heat maps are likely to occur in practice and are therefore informative to the classifier.

Of the various distortions one may apply to images, the domain-specific ones we consider are zooming in/out, rotating, and shearing. (Translating also applies, but ConvNets are invariant to translations.) Through trial and error, we notice that zooming does not significantly help or hurt classification, so our augmentation scheme is to rotate heat maps by an angle randomly chosen in the range ±20◦, and then shear them by an angle randomly chosen in the range ±10◦. We engineer ten random distortions for each of the 697 heat maps, increasing the size of the data set to 7667 (697 + 6970). The exact counts of each treatment site are shown in Table 1. By chance, some distortions are barely noticeable, and others, obvious, as illustrated in Figure 7.

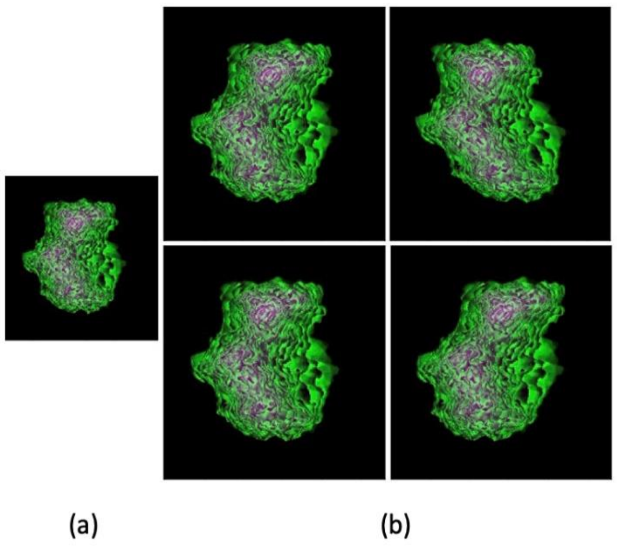


Figure 7: (a) Original prostate plan heat map and (b) four random distortions for data augmentation.

4. Results and Discussion

This section is an evaluation of how well our heat maps capture information indicative of treatment site. To this end, we choose to employ deep-learning ConvNets because they are classifiers that can accept images at their inputs. Evaluation is conducted on several variants of two ConvNets popular for image classification, namely VGG [18] and ResNet [19], because they have previously been successfully applied to the medical field [20,21] and are readily available in standard PyTorch framework. We describe the changes we make to their standard architectures to adapt to our specific data set. Then, we report the classifiers' testing accuracy, confusion, and F-1 score and find that the classifiers perform very well, even outperforming our previous benchmark [14,15].

4.1. Models

4.1.1. Architectures: The details of convolution, pooling, and dense layers that typically comprise a ConvNet architecture will not be covered here, but we encourage the reader to review any of the many works in the literature, for example [22]. Convolution and pooling layers usually occur first within the architecture's layer chain. These layers make up the feature module that learns features like edges, curves, and angles, for example. Importantly, the feature module is invariant to translation, so a heat map looks the same to the architecture even if it has been shifted in the image. Typically preceded by a flatten layer and occurring at the end of the architecture's chain, the classifier module consists of dense layers and learns the relationship between the learned features and the target of interest—in our case, the treatment site.

The VGG-16 architecture consists of a total of 16 convolution and dense layers. A variant is the VGG-19, which supports three additional convolution layers. VGG architectures have been shown to perform very well for many object-recognition tasks with a variety of data sets and applications including medical imaging [21]. ResNets, another class of ConvNets, introduce short-circuiting into the layer chain, providing parallel paths for data to bypass selected layers in the chain. Bypass connections may improve classifier performance because they allow for the passing of low-level information from layers early in the chain directly to deeper layers, so they see both low- and high-level information. Like VGG, ResNets come in many architectures of varying depth including ResNet-18, -34, -50, -101, and -152. Also, like VGG, ResNet architectures have been shown to perform well in many different situations, including medical imaging [20].

4.1.2. Transfer Learning: We avoid training the classifiers from scratch by loading models previously trained on ImageNet [23]. Then, we freeze the feature module's parameters and train only the classifier module. This process, known as transfer learning, is advantageous for two reasons: First, the computational effort required to train the classifiers is significantly reduced because the number of trainable parameters is only a small fraction of what it would be otherwise. Second, our relatively small data set of is insufficient to train such deep networks from scratch and expect good performance. Transfer learning works in part because ImageNet is an expansive, feature-rich data set. The low-level features the models previously learned to discern images in ImageNet are applicable to classifying our heat maps as well.

We strip the classifier module of VGG-16 and replace it with two dense layers which we train with the heat map data set. The first dense layer is a 512-by-512 fully connected layer with Rectified Linear Unit (ReLU) activation and a dropout rate of 0.2 to guard against overfitting. The subsequent dense layer is a 512-by-3 fully connected layer with softmax activation for yielding a vector of three probabilities, one for each class. We do the same for the VGG-19 architecture.

Similarly, for ResNet-18 and -34, we strip away their respective classifier module and replace it with the same two dense layers as with the VGG. ResNet-50, -101, and -152 are handled a bit differently due to their different architectures preceding the classifier module. For these architectures, our first dense layer is 2048-by-512 in shape rather than 512-by-512. Everything else (second dense layer, ReLU, dropout, softmax) remains the same.

4.1.3. Scaling Considerations: As each of the architectures was pre-trained on the ImageNet data set, they each expect pixel values of the same scale as ImageNet pixels. As such, we scale the red channel pixels to have (mean, deviation) of (0.485, 0.229), the green channel (0.456, 0.224), and the blue channel (0.406, 0.225) in accordance with [23].

4.2. Classification Results

All seven classifiers (the two VGG variants and five ResNet variants) are each separately evaluated within its own five-fold cross-validation scheme. Data are split into training and testing partitions in such a way that each augmented image accompanies its respective original image in the same partition. Data are scaled only after the train-test split. Both measures are taken to preserve the integrity of the cross-validation process.

We use cross-entropy as the loss function [24] for training and record the training loss versus epoch, where one epoch is one complete pass through the training set in batch gradient descent. We also track the training and testing accuracies versus epoch, which help us decide the total number of epochs for training. After some trial and error, we choose to terminate training after 50 epochs because that is generally when we see a trending decline in testing accuracy—a sign that the model begins overfitting the training data. Up to the 50th epoch, we generally observe an asymptotic decline in training loss and (nearly) no decrease in testing accuracy. We repeat this process separately for each fold and for each of the models tested and observe similar behavior among all five folds and seven models.

	VGG-16	VGG-19	ResNet-18	ResNet-34	ResNet-50	ResNet-101	ResNet-152
Accuracy	94.5%	91.1%	**97.8%**	96.7%	94.5%	92.4%	90.2%
F-1 score	0.947	0.911	**0.979**	0.967	0.946	0.924	0.902

Table 3: Mean accuracy and F-1 score of each model.

4.2.1. Accuracy: Accuracy, the simplest performance metric, is the percentage ratio of correct predictions to total predictions. With five-fold cross-validation, we arrive at five accuracies—one for each test set—and, in the first row of Table 3, report the mean accuracy across the five folds. According to the table, ResNet-18 performs the best with a mean accuracy of 97.8%, and the deeper ResNet architectures do not perform as well. With an accuracy of 94.5%, the best-performing VGG architecture, VGG-16, does not perform as well as ResNet-18.

4.2.2. Confusion Matrix: Accuracy, although simple and convenient, does not consider class information and thus lacks in detail. This point is especially important given the class-imbalance of our data set, for which we have about 64% more breast plans than head-neck plans. We therefore report a more insightful metric, a confusion matrix, which is a contingency table of actual-versus-predicted tallies.

Since each treatment plan in a cross-validation scheme is used as a test plan exactly once, the confusion from each of the five folds may be tallied together to provide a single confusion matrix for each model. Table 4 shows the confusion of the ResNet-18 model, and we notice that breast plans are the easiest for the model to classify. All the treatment plans that are actually breast plans are identified as such, and no head-neck or prostate plans are predicted as breast. 97.8% of plans that are actually head-neck are predicted correctly; the other 2.2% are predicted as prostate plans. Similarly, 94.7% of prostate plans are correctly identified, and the other 5.3% of them are predicted to be head-neck plans.

For brevity, we only report the confusion of ResNet-18, since it is the model with the highest accuracy and seems to perform the best. However, we note that the performances of other models are like ResNet-18, but with more confusion. That is, all the models can identify the breast plans perfectly, just as ResNet-18. The other models show an increased rate of misclassifying head-neck and prostate plans when compared to ResNet-18.

		Predicted			
		Head-neck	**Breast**	**Prostate**	**Total**
Actual	**Head-neck**	**2002**	0	44	2046
	Breast	0	**3355**	0	3355
	Prostate	121	0	**2145**	2266
	Total	2123	3355	2189	7667

Table 4: Confusion matrix for the ResNet-18 model.

4.2.3. F-1 Score: An F-1 score is a convenient way of representing confusion as a single number and is preferable when classes are imbalanced, as with our data set. An F-1 score considers both the precision and recall, describing the model's balanced ability to capture true positives and simultaneously avoid false positives. It is the harmonic mean of the precision and recall, and a perfect classifier has an F-1 score of 1. For each model, we calculate the F-1 score by class, and then perform a weighted average over each class to obtain the final F-1 score reported. Using Table 4 for ResNet-18, for example, the precision, recall and F-1 score for the head-neck class are 2002/2123 = 0.943, 2002/2046 = 0.978, and $\sqrt{0.943 \cdot 0.978} = 0.961$ respectively. Similarly, the breast and prostate classes have F-1 scores of 1 and 0.963, respectively. Thus, the class-weighted F-1 score for ResNet-18 is (0.961(2046) + 1(3355) + 0.963(2266))/7667 = 0.979. In the last row of Table 3, we report the class-weighted F-1 scores for each model, and again we see that ResNet-18 is the best. The model's architectural details are provided in [25].

4.3 Computational Effort: Classification

We evaluate all classifiers on a different machine from the one on which heat maps were generated. Classifiers are trained and tested on a single Apple iMac machine with a 3.6-GHz 10-Core Intel Core i9 processor and 64 GB 2667 MHz DDR4 memory. Using Spyder, we run PyTorch with Python 3.9, utilizing the machine's AMD Radeon Pro 5300 4 GB

GPU. Each model requires about the same amount of time for training—about 8 hours for our specific choices of training size and number of training epochs.

Making predictions on a test set is relatively quick, requiring only about one minute. Rigorously evaluating a model in a five-fold cross-validation scheme, which necessitates training the model five times, consequently requires about 40 hours to complete.

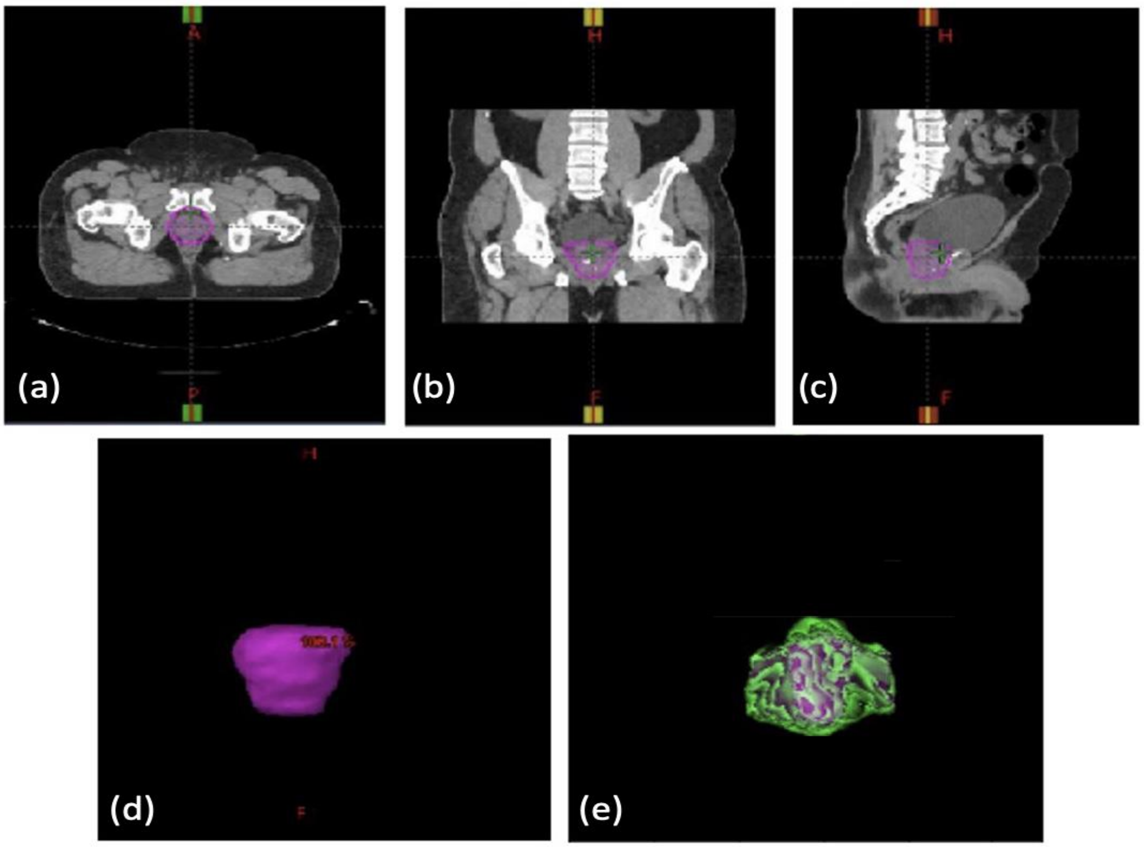


Figure 8: One misclassified prostate cone-down boost plan. (a) Axial, (b) coronal, (c) sagittal views from CT scan with target outlined and shaded in pink. (d) 3-D view of target from CT scan, (e) plan's heat map generated by our method.

4.4 Discussion

A plan's heat map is a two-dimensional representation of the plan's spatial dose distribution leaving the LINAC. Though not identical to the shape of the cancer, which is instead three dimensional and potentially calculated from additional geometry data of the MLC relative to the patient, a heat map's shape is closely related to that of the cancer to which it corresponds. This was the motivating factor for structuring the data this way, since we believe cancers of the same treatment site are each similarly shaped but dissimilar to shapes of cancers of different treatment sites.

Still, the heat maps contain enough treatment plan information for deep-learning classifiers to automatically verify a plan's treatment site. This fact is made clear by the outstanding performance measures of each classifier evaluated, particularly ResNet-18. The classifiers are especially accurate at discerning breast plans, most likely due to their distinctive rectangular shape lacking curves and gradients like in Figures 3 and 6. On the other hand, classifiers can confuse prostate and head-neck plans because they typically share similar features like in Figure 3.

Including site fractions s (Section 3.2.4) into the data structure improves the classifier performance, indicating the importance of this piece of data. Originally during experimentation, site fractions were omitted from heat map construction and only single-channel (grayscale) heat maps fed the ML classifier immediately after the filtering step in Figure 4. In that case, the best-performing model, once again ResNet-18, achieved an accuracy and F-1 score of only 95.3% and 0.950, respectively. Compared to 97.8% and 0.979, both accuracy and F-1 score increased more than 0.02 with the inclusion of site fractions. In fact, comparison of the confusion matrix without site fractions with Table 4 reveals that site fractions help differentiate prostate plans from head-neck. Indeed, the fractional dose distribution of many prostate plans is like those of head-neck but different in total dose, and vice-versa—at least for our data set. The inclusion of s helps the classifier in these cases.

Given the complexity of the raw treatment plan data, it is no small feat to identify optimal information useful for plan checking. The heat-map structure, with its inclusion of site fractions, is the best representation for treatment site identification we know of. A sub-optimal method, Bai and Xia [16] used only the variances of leaf width positions and LINAC gantry angle as features rather than heat maps. We found the accuracy of applying their method to our data set was a modest 79.9%, far below the accuracy achieved with our heat maps. (F-1 score was similar.) Moreover, the heat maps in this current work are an improved iteration of heat maps in our previous work in [15], which omitted leaf widths and MLC rotation angles. The improved accuracy and F-1 score of the current method over the previous imply the usefulness of leaf widths and MLC angle towards automated plan checking. This makes sense since mechanical design of the LINACs including leaf widths is presumably already optimized by machine vendors.

Table 4 shows that the best-performing model, ResNet-18, misclassifies 165 plans out of 7667. Interestingly, manual inspection of the misclassified plans reveals that they are mostly (87%) cone-down boost plans like the prostate plan in Figure 8. In practice, these types of plans are concentrated and localized, usually employing a small aperture. For example, a patient's initial plan in a sequence of plans may involve treating the prostate and surrounding nodes, but the second plan in the sequence may only treat the prostate, presumably with a smaller aperture. The second plan in this case is a cone-down boost plan. It makes sense that the classifiers confuse these plans—they are residuals of a larger scope of treatment, and our classifiers are not provided the important sequential information.

Clearly, including more layers into the architecture is not helpful at classifying the treatment site. This is often the case because the last few layers in deeper networks see almost no information about the original image [26]. In fact, the ResNet architecture was designed to address this issue. We believe the results in Table 3 indicate such an issue because (1) the performance drops as the models become deeper, and (2) the ResNet models perform better than the VGG models. We investigate further by observing the filter activations as the images are passed forward through the trained architecture. For each of eight separate heat maps, Figure 9 shows the output of a pre-selected filter in the second layer and one in the fourth layer of the ResNet-18 model. From the figure, we observe quite a reduction in information between the two layers, and very little information present even after the fourth layer. The figure makes it clear that adding more layers will not necessarily improve classifier performance. It is also worth noting the success of transfer learning. Pre-trained on ImageNet, the filters learned features applicable to our heat maps, for example, edges directed top-left to bottom-right as in Figure 9(a).

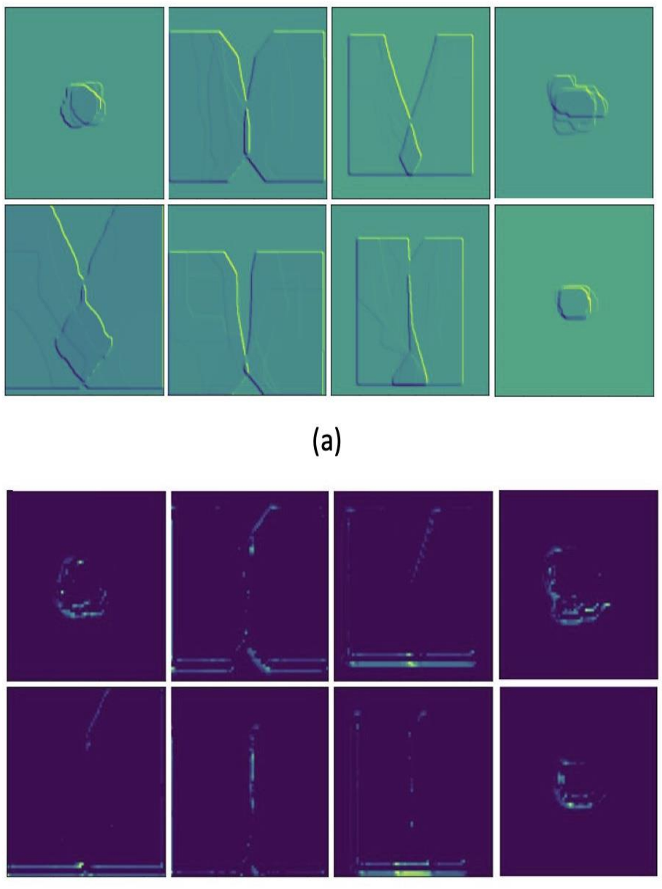


Figure 9: (a) The output of a single filter in the second layer of ResNet-18 for each of eight separate input heat maps, and

(b) the output of a filter in the fourth layer for the same eight heat maps.

Data augmentation is a useful technique and plays an important role in our study. We find the technique boosts the accuracy of the ResNet-18 model almost 1%, from 97.1% to 97.8%, but at the cost of 1000% more computational time (four hours versus 40). Care must be taken to distort the images in a way that makes sense for the specific application domain. For our application in which images are heat maps of radiation distribution, some distortions like additive noise do not apply.

Treatment plan data were curated from two separate clinics operating LINACs from two different vendors, which implies the independence of our method from treatment process implementation details. Still, it is possible the two clinics in our study are, by chance, like one another, and that our methods will not fare so well with data from other clinics—a limitation to our study. It is true that every clinic each operates with its own unique nuance. Yet at all clinics, treatment sites are the same, and the governing treatment practice is standard. Therefore, is unlikely the case that our methods will not generalize to other clinics.

5 Conclusion

5.1. Remarks

Owing to increased complexity in radiation therapy treatment technology, computer errors arise in treatment planning and have previously resulted in severe patient harm when radiation is delivered to an unintended treatment site. This study established a deep-learning-based approach to detect such errors by structuring treatment plan data as a heat map and investigating the ability of several convolutional neural networks to classify a plan's treatment site given its heat map. A plan's heat map contains enough information for the ResNet-18 architecture to predict the plan's treatment site with a near perfect accuracy rate of 97.8% and an impressive F-1 score of 0.979. Data augmentation and transfer learning methods were successfully applied to train the deep networks, implying that smaller clinics with modest data sets can apply our method in practice. It is reassuring that machine-learning algorithms can be effectively applied to safeguard radiation treatment.

The results are encouraging, but the study was limited to only head-neck, breast, and prostate cancers. More types of cancer treatment plans must be collected, analyzed, and studied to demonstrate the robustness of the method. In this case, the method can easily be extended. Of the three treatment sites considered, head-neck and prostate plans were the most challenging to classify, and an overwhelming majority of misclassified plans are cone-down boost plans. Future research can use this insight to possibly improve the classifier performance. For example, assigning cone-down boost plans their own separate class may improve the results. Alternatively, since a cone-down boost plan is one plan in a sequence of plans, recurrent neural networks may be applied to learn the sequential information.

In addition, we plan to integrate our heat maps and the deep-learning classifier with ChartAlert (Infondrian LLC„ Iowa city, IA), a cloud-based software to automate the detection of treatment errors in radiation oncology. By seamlessly integrating our methods into ChartAlert using an Application Programming Interface (API), clinical care teams can access real-time predictions for treatment site verification. This integration can reduce treatment errors and improve patient outcomes.

To validate the classification accuracy and clinical utility in a broader patient population, we will collaborate with other healthcare institutions to collect more diverse and representative data. By integrating our methods into clinical decision support systems, we can revolutionize treatment site verification and improve patient outcomes across various healthcare settings.

5.2 Acknowledgements

Research reported in this publication was supported by National Cancer Institute of the National Institutes of Health under award number R42CA195819.

Conflict of Interest

The authors declare that they have no conflict of interest.

References

1. Vijayakumar S, Duggar WN, Packianathan S, et al. (2019) Chasing zero harm in radiation oncology: Using pre-treatment review peer review. Frontiers Oncol 9: 302.
2. Gopan O, Zeng J, Novak A, et al. (2016) The effectiveness of pretreatment physics plan review for detecting errors in radiation therapy. Med Phys 43(9): 5181-5187.
3. Ericson J (2014) Death rays. Newsweek Magazine.
4. Bogdanich W (2010) The radiation boom—Radiation offers new cures, and ways to do harm. The New York Times.
5. Akashi M, Cosset JM, Gourmelon P, et al. (2001) Investigation of an accidental exposure of radiotherapy patients in Panama. Intl Atomic Energy Agency.
6. Jin D, Sergeeva E, Weng W, et al. (2022) Explainable deep learning in healthcare: A methodological survey from an attribution view. WIREs Mechanisms of Disease 14(3).
7. Mayo CS, Kessler ML, Eisbruch A, et al. (2016) The big data effort in radiation oncology: Data mining or data farming? Adv Rad Oncol 1(4): 260-271.
8. Davenport T, Kalakota R (2019) The potential for artificial intelligence in healthcare. Future Health J 6(2): 94-98.
9. Dhillon A, Singh A (2021) Machine learning in healthcare data analysis: A survey. J Biol Today's World 8(6).
10. Boyer A, Biggs P, Galvin J, et al. (2001) Basic Applications of multileaf collimators. Technical report. Med Phys Publishing.
11. Azmandian F, Kaeli D, Dy JG, et al. (2007) Towards the development of an error checker for radiotherapy treatment plans: a preliminary study. Phys Med Biol 52(21): 6511-6524.

12. Kisling K, Cardenas C, Anderson BM, et al. (2020) Automatic verification of beam apertures for cervical cancer radiation therapy. Pract Radiat Oncol 10(5): e415-e424.
13. Kalet AM, Gennari JH, Ford EC, et al. (2015) Bayesian network models for error detection in radiotherapy plans. Phys Med Biol 60(7): 2745-2749.
14. Kump PM, Xia J, Yaddanapudi S, et al. (2022) An automated treatment plan alert system to safeguard cancer treatments in radiation therapy. Presented at Canadian Org Med Phys (COMP) Ann Sci Meeting, Quebec City, Quebec, Canada.
15. Kump PM, Xia J, Yaddanapudi S, et al. (2023) An automated treatment plan alert system to safeguard cancer treatments in radiation therapy. Machine Learn Appl 10: e100437.
16. Bai E, Xia J (2021) A knowledge based automatic radiation treatment plan alert system. Intl J Artifl Intel & Appl 12(6).
17. Lo J, Cardinalla J, Costanzo A, et al. (2021) Medical augmentation (med-aug) for optimal data augmentation in medical deep learning networks. Sensors 21(21): 7018.
18. Simonyan K, Zisserman A (2015) Very deep convolutional networks for large-scale image recognition. Intl Conf Learn Representation.
19. He K, Zhang X, Ren S, et al. (2015) Deep residual learning for image recognition. arXiv.
20. Lu S, Wang SH, Zhang YD (2020) Detecting pathological brain via ResNet and randomized neural networks. Heliyon 6(12): e05625.
21. Yadav SS, Jadhav SM (2019) Deep convolutional neural network based medical image classification for disease diagnosis. J Big Data 6(113).
22. Li Z, Liu F, Yang W, et al. (2015) A survey of convolutional neural networks: Analysis, applications, and prospects. IEEE Trans Neural Netw Learn Syst 33(12): 6999-7019.
23. Deng J, Dong W, Socher R, et al. (2009) ImageNet: A large-scale hierarchical image database. IEEE Conf Comp Vision Pattern Recognition: 248-255.
24. Andreieva B, Shvai N (2021) Generalization of cross-entrop loss function for image classification. Mohyla Mathematical J 3: 3-10.
25. Rmzan F, Khan MU, Rehmat A, et al. (2019) A deep learning approach for automated diagnosis and multi-class classification of Alzheimer’s disease stages using resting-state fMRI and residual neural networks. J Med Sys 44(2): 37.
26. Ayyadevara, VK, Reddy Y (2020) Modern Computer Vision with PyTorch. Packt Publishing.

***Corresponding author:** Paul M. Kump, PhD, Department of Electrical and Computer Engineering, College of Engineering, Kansas State University, Manhattan, KS, 66506, USA, Tel: (718) 409-3351; Email: kump@ksu.edu

Received date: July 04, 2023; **Accepted date:** August 17, 2023; **Published date:** August 22, 2023

Citation: Kump PM, Xia J, Yaddanapudi S, Bai E (2023) A Deep-Learning Error Detection System in Radiation Therapy. *Ann Biomed Res* 5(1): 126.