UNIVERSIDAD DE CHILE
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS
DEPARTAMENTO DE INGENIERÍA ELÉCTRICA

SEGMENTACIÓN SEMÁNTICA PARA ALIMENTOS

MEMORIA PARA OPTAR AL TÍTULO DE INGENIERO CIVIL ELÉCTRICO

PATRICIO ANTONIO FLORES VIDELA

PROFESOR GUÍA:
IVÁN CASTRO OJEDA

MIEMBROS DE LA COMISIÓN:
FRANCISCO RIVERA SERRANO
ANDRÉS THOMAS BRIGNETI

SANTIAGO DE CHILE
2023

SEGMENTACIÓN SEMÁNTICA PARA ALIMENTOS

El presente documento corresponde a la memoria del autor, centrada en la segmentación semántica para alimentos, particularmente sobre ingredientes típicos de pizzas. Dicha misión es encomendada por Kwali S.P.A., empresa dedicada a la supervisión de procesos productivos, enfocada particularmente en franquicias de comida rápida (pizzerías).

En este documento se motiva y detalla el proceso de implementación de un modelo de segmentación semántica especializado para realizar la tarea descrita, junto con los objetivos del proyecto y el contexto sobre los tópicos más relevantes asociados al procesamiento de imágenes. Además, se exponen los estados del arte de la segmentación semántica (un algoritmo supervisado "FoodSeg" y otro no supervisado "STEGO"), sumado a un tercer algoritmo (supervisado) recomendado por la empresa (DeepLabv3). Dichos algoritmos son implementados y validados visualmente sobre el problema especificado, en donde se tienen limitantes (baja cantidad de datos etiquetados) para realizar un correcto reentrenamiento sobre el estado del arte supervisado. Con esto, son seleccionados los mejores modelos resultantes (STEGO y DeepLabv3) para ser reentrenados y evaluados con diferentes métricas de medición (especificadas en el documento), con lo cual, se obtiene que el algoritmo recomendado (DeepLabv3) presenta los mejores resultados, siendo capaz de segmentar de mejor manera según las métricas, respecto a los demás algoritmos mencionados. En donde, debido a la limitada cantidad de datos etiquetados con los que fue entrenado DeepLabv3, se cree que este presentará una mejora en sus resultados al ser reentrenado con una mayor cantidad de datos etiquetados.

A mi querida familia,
gracias por todo el apoyo.

Patricio

Tabla de Contenido

Índice de Tablas

Índice de Ilustraciones

Capítulo 1

Introducción

El presente documento corresponde a la memoria sobre segmentación semántica para alimentos, en la cual se pueden encontrar los capítulos: (1) Introducción, en donde se dará el contexto necesario para comprender el desarrollo de la memoria, identificando y formulando el problema, junto con los objetivos buscados; (2) Marco teórico y estado del arte, donde se expondrán los fundamentos teóricos para comprender el documento, las dos mejores alternativas de solución propuestas previamente por otros y un algoritmo recomendado por Kwali; (3) Desarrollo, en donde se expondrán los datos utilizados, las pruebas más relevantes y sus evaluaciones visuales; (4) Resultados obtenidos con el o los mejores modelos resultantes; (5) Análisis, en donde se analizarán y se comparan los resultados expuestos en la sección anterior. Finalmente (6) Conclusiones, en donde se expondrá las conclusiones y aportes finales del trabajo realizado.

La presente memoria está centrada en la segmentación semántica de alimentos haciendo uso de visión computacional (de lo cual se puede encontrar mayor información en el marco teórico). Dicha problemática es encomendada por Kwali Chile S.P.A., empresa en la cual se desarrolla la misma.

Kwali está dedicada a la supervisión de calidad de alimentos para franquicias de comida rápida mediante visión computacional, específicamente, centrándose en pizzerías de diferentes cadenas dentro de Estados Unidos, Chile y Panamá. La empresa está avanzando para lograr estimar la cantidad (en masa) de los ingredientes que componen dichas pizzas, logrando estimar así los costos de producción de cada una de estas, pudiendo indicar con esta información a las franquicias cómo reducir sus costos de producción (al evitar el uso excesivo de ingredientes) lo que se verá reflejado directamente en las ganancias de las mismas. Por lo cual, y para avanzar en el objetivo recién expuesto, en esta memoria se busca lograr una correcta segmentación semántica de alimentos, específicamente, sobre pizzas y sus diferentes tipos de ingredientes. Para esto, se trabajará con inteligencia computacional en el área de procesamiento de imágenes.

1.1. Identificación y Formulación del Problema

Trabajar con procesamiento de imágenes dentro del sector alimentario, requiere características específicas para reconocer las diferentes propiedades de los alimentos como, por ejemplo, su madurez, diferentes tamaños de un mismo alimento, diferentes formas al ser

cortados, posibles defectos, grados de cocción, entre otros.

Las técnicas que se encuentran en la bibliografía ("FoodSed" [3] modelo supervisado y "STEGO" [4] no supervisado) están creadas o implementadas para conjuntos de datos, públicamente disponibles (*COCO-Stuff*, *Cityscapes*, *FoodSeg103*), cuyos dominios no siempre corresponden al dominio de los productos alimentarios. En particular, "FoodSed" al ser un modelo supervisado, aprende a identificar específicamente las clases con las que es entrenado, en donde se tiene la clase "pizza" como un único objeto. Por su parte, si bien "STEGO", es no supervisado, este igual es entrenado previamente para identificar pizzas completas como un único objeto. Por esta razón, los modelos de segmentación semántica disponibles, no se adaptan correctamente a los alimentos (ingredientes de pizzas y sus variadas preparaciones) con los que trabaja Kwali, sino que, como se mencionó (y puede profundiza en el capítulo 3), estos modelos son entrenados para segmentar pizzas como una sola clase, sin poder realizar la segmentación de los ingredientes que las componen.

Específicamente, el problema que se busca resolver es la correcta segmentación semántica de alimentos presentes en pizzas; esto para las diferentes franquicias que contratan los servicios de Kwali. Con esto, se busca poder identificar tanto la posición como los tipos de ingredientes en imágenes de pizzas, los cuales pueden variar dependiendo de cada franquicia.

1.2. Objetivos del Trabajo de Título

En esta memoria se tiene como objetivo general implementar al menos un modelo de segmentación semántica para alimentos que sean capaz de identificar los diversos ingredientes que se pueden encontrar, típicamente, en la preparación de pizzas. Estos ingredientes son: queso, pepperoni, albóndigas, tomate, pollo, carnes rojas, vegetales verdes (espinaca y pimentón verde), aceitunas y la masa del borde de la pizza.

Con lo anterior, se tiene como objetivos específicos:

- Encontrar e implementar algún o algunos algoritmos útiles para el problema de segmentación a partir de una red profunda u otro tipo de arquitectura preentrenada, la cual haya sido específicamente creada para el problema de segmentación, mas no necesariamente para la segmentación semántica de alimentos.

- Validación visual del(los) algoritmo(s) anterior(es) para la segmentación semántica de alimentos típicos en pizzas (ingredientes explicitados en el objetivo general), siendo capaz de delimitar e identificar los píxeles correspondientes a los diferentes ingredientes presentes en diferentes tipos de pizzas con los que trabaja Kwali. En caso de obtener resultados poco satisfactorios, se buscará reentrenar y/o modificar dichos algoritmos (ya sea en su estructura o en sus parámetros, lo que podría ser: incluir o eliminar capas, modificar las salidas o su tamaño, cambiar tasas de aprendizajes, tamaño de los batches, el número de pasos de entrenamiento o las épocas de entrenamiento).

- Evaluar el rendimiento del o los algoritmos propuestos respecto a diferentes tipos de pizzas con los que trabaja Kwali y sus diferentes tipos de ingredientes, los cuales pueden cambiar para el mismo tipo de pizzas dependiendo la franquicia de origen. Se busca obtener al menos un sesenta por ciento (60 %) de acierto en la métrica *precision*, al

menos un noventa por ciento (90 %) de *accuracy* y por lo menos un cuarenta por ciento (40 %) en la métrica IoU, esto con fin de resolver el problema especificado.

- Finalmente, se realizará un análisis y comparación entre resultados de los algoritmos utilizados con y sin el entrenamiento específico para el problema descrito.

Es importante notar que se utilizarán diferentes métricas de medición para comparar los resultados, sin embargo hay que destacar que hay métricas más relevantes que otras (dada la necesidad de Kwali). Teniendo en cuenta esto, se consideran relevantes las explicitadas con anterioridad.

Se dice que IoU y *precision* son más significativas, dado que consideran los aciertos correctos del modelo (verdaderos positivos o TP) por sobre el resto de las posibilidades (verdadero negativo, falso negativo y falso positivo), siendo más relevante para Kwali pues, les es más importante tener la certeza de detectar correctamente un ingrediente, antes que detectar correctamente su ausencia (verdadero negativo) lo que no aporta mucha más información. Sin embargo, para la medición del modelo, también se considera la información de las fallas, pero con una importancia menor, teniendo el siguiente orden de relevancia: TP > FP > FN > TN. Luego, como se puede observar en la sección de marco teórico, las métricas que más importancia le dan a los TP son *precision* y IoU, aunque, considerando que, si bien aporta menos información que el resto, no se debe omitir la información dada por los TN. Por esto, se opta por utilizar también *accuracy* como medida, más no medida principal. Se puede encontrar mas información en la sección de análisis.

Capítulo 2

Marco Teórico y Estado del Arte

2.1. Marco Teórico

A continuación, se expondrán los fundamentos teóricos de los elementos más importantes relacionados a la memoria.

2.1.1. Segmentación Semántica

Se entiende por segmentación semántica a la acción de asignar una determinada etiqueta o categoría a cada uno de los píxeles de una imagen, detectando y reconociendo los diferentes objetos dentro de la misma, segmentando las fronteras de los objetos encontrados de manera exacta o casi exacta y midiendo su desempeño respecto a la cantidad de píxeles correctamente etiquetados.

Para realizar la segmentación de los objetos deseados, típicamente, es necesario contar con dos sub bloques en serie dentro del o de los algoritmos a utilizar, los cuales son: un bloque de extracción de características o extractor de características y un bloque de clasificación o clasificador que haga uso de las características extraídas previamente por el bloque anterior, obteniendo así, rasgos que permitan al clasificador diferenciar los píxeles entre las diferentes clases existentes para su correcta segmentación. En los puntos "2.1.3. Extracción de características" y "2.1.5. Clasificación" de la presente sección, es posible encontrar qué es un extractor de características y qué es clasificación, respectivamente.

2.1.2. Visión Computacional

La visión computacional también llamada visión artificial, es el área de la inteligencia computacional centrada en la extracción de información de determinadas propiedades de una imagen (o conjunto de estas), logrando verificar un proceso o actividad específica. Esta tecnología es capaz de establecer parámetros en procesos productivos haciendo uso de computadoras y cámaras de vídeo, analizando e interpretando el contenido de las imágenes entregadas al sistema, simulando así la inspección humana, permitiendo a un sistema automático tomar diversas decisiones dependiendo del problema a enfrentar [20]. En nuestro caso, el problema es la delimitación e identificación de alimentos.

2.1.3. Extracción de Características

Para realizar la correcta clasificación de los píxeles dentro de la imagen a tratar, es necesario extraer información característica o "única" de dichos píxeles, tal que, dicha información permita la diferenciación entre las posibles clases a escoger, en donde, a mayor robustez del sistema de extracción de características, mayor o mejor podrá ser el desempeño del sistema que haga uso de dichas características [2].

Comúnmente, los extractores de características más utilizados son las redes neuronales convolucionales (CNN por sus siglas en inglés), sin embargo, en los últimos años estas CNN se han ido adaptando a los Transformadores de Visión ($Vision\ Transformers$ o ViT) dentro de sus arquitecturas y hasta se ha demostrado que un transformador puro, aplicado directamente a secuencias de parches de imagen, puede funcionar muy bien en tareas de clasificación de imágenes [6]. Para este proyecto, se usan ambos tipos de extractores de características para obtener información relevante de las imágenes de pizzas por delimitar.

2.1.4. Aprendizaje Supervisado y no Supervisado

Los algoritmos de aprendizaje supervisado basan su aprendizaje en un conjunto de datos de entrenamiento previamente etiquetados. Por etiquetado entendemos que, para cada ocurrencia del conjunto de datos de entrenamiento, conocemos el valor de su atributo objetivo.

Por otra parte, los algoritmos no supervisados, son algoritmos que basan su proceso de entrenamiento en un conjunto de datos sin etiquetas o clases previamente definidas. Es decir, a priori no se conoce ningún valor objetivo o de clase, ya sea categórico o numérico [10].

Para este proyecto se hace uso de ambos tipos de aprendizajes, los cuales son utilizados por diferentes algoritmos de los que se pueden encontrar mayor información en la sección de estado del arte.

2.1.5. Clasificación

La clasificación automática consiste en utilizar técnicas de Inteligencia Artificial sobre un conjunto de elementos para ordenarlos ya sea por clase o por categoría, pudiendo utilizarse estas técnicas para asignar un elemento a una determinada clase o categoría [11]. Cabe destacar, que existen enfoques de clasificación basados en aprendizaje supervisado y en aprendizaje no supervisado.

2.1.6. Redes Neuronales

Las redes neuronales artificiales consisten en un conjunto de unidades llamadas neuronas artificiales, las cuales están inspiradas en neuronas biológicas. Dichas neuronas artificiales son ordenadas por capas y conectadas a otras capas de neuronas de manera secuencial, en donde cada neurona tiene un peso y la salida de estas es la combinación lineal de sus entradas junto con una función de activación (pudiendo ser una función limitadora o umbral) que, posteriormente, se utiliza como entrada de las neuronas en la siguiente capa. Esto, para transmitir información entre las capas, intentando simular así el funcionamiento de un cerebro biológico al procesar información [8].

Por otro lado, una red neuronal convolucional (*Convolutional Neural Network* o CNN por sus siglas en inglés) es un tipo de red neuronal artificial en donde las neuronas se ven sometidas a una operación de convolución a su salida. Esto, dado por un núcleo convolucional

correspondiente a la respectiva neurona. Este tipo de red es una variación de un perceptrón multicapa [15], sin embargo, debido a que su aplicación es realizada en matrices bidimensionales, es muy efectiva para tareas de visión artificial, como en la clasificación y segmentación de imágenes, entre otras aplicaciones [9]. Esto debido a la operación de convolución, con la cual es posible extraer información de la imagen dada (al realiza un filtrado de los valores de píxel de dicha imagen, pudiendo detectar ejes o realces, entre otras cosas.). Sin embargo, al realizar muchas operaciones de convolución normales, estas terminan por reducir la dimensionalidad del objeto de entrada y por tanto, su información.

2.1.7. Vision Transformer (ViT)

Los transformadores de visión (ViT por sus siglas en inglés) son un modelo para la clasificación de imágenes que emplea una arquitectura similar a la de un transformer [7] sobre parches o secciones de una imagen.

A grandes rasgos, los ViT proyectan linealmente parches aplanados en un espacio de menor dimensión respecto a la de la imagen de entrada. Posteriormente a la proyección, se agregan "incrustaciones" (*embedding*) aprendidas respecto a la posición de los parches, en donde el ViT es capaz de aprender a codificar la distancia dentro de la imagen respecto a la similitud de las incrustaciones de posición, es decir, los parches más cercanos tienden a tener incrustaciones de posición más similares. Finalmente, la "auto atención" ($Self\text{-}attention$) permite al ViT integrar información de la imagen dada. Esto, a lo largo de toda su estructura, inclusive en las capas más bajas [6].

Como se mencionó con anterioridad, los ViT son posibles de implementar como un extractor de características, tal como se puede ver en la sección de estado del arte.

2.1.8. Conditional Random Fields (CRF)

Los campos aleatorios condicionales o CRF, por sus siglas en inglés, son una clase de métodos de modelado estadístico que a menudo se aplican en el reconocimiento de patrones, en el aprendizaje automático y se utilizan para la predicción estructurada. Mientras que un clasificador convencional predice una etiqueta para una sola muestra (sin tener en cuenta las muestras vecinas), un CRF es capaz de tener en cuenta el contexto. Para ello, las predicciones se modelan como un modelo gráfico que representa la presencia de dependencias entre las predicciones [17].

Por lo anterior, un CRF es un clasificador útil para ser aplicado sobre el problema de segmentación planteado anteriormente, siendo utilizado como bloque de salida en STEGO, algoritmo expuesto en la siguiente sección de estado del arte.

2.1.9. Tensores

Los tensores son generalizaciones de matrices (multidimensionales) para datos de orden superior, es decir, matrices que contienen matrices. Estos proporcionan formatos de representación de datos realmente útiles, los cuales son utilizados en muchos dominios, incluyendo la estadística, ciencia de datos y aprendizaje automático.

Los tensores son utilizados, por ejemplo, en sistemas sensibles al contexto, en donde se presentan diferentes patrones que pueden ser modelados de manera efectiva por un tensor de n-ésimo orden [21]. Para usos del presente documento, los tensores son utilizados para

contener la información extraída por los extractores de características y con los cuales se pueden hacer operaciones, de forma similar a las matrices. Estos son mencionados en el estado del arte, al describir cómo se extrae información de imágenes.

2.1.10. Datasets

Los *dataset*, también conocido como conjuntos de datos o set de datos, son, como bien dice su nombre, conjuntos o colecciones de datos, los que generalmente son tabulados y habitualmente ordenados para su uso en diferentes aplicaciones, no obstante estos pueden ser también una colección de documentos o de archivos. En particular, en este documento se hablará de *datasets* haciendo referencia a conjuntos de imágenes, específicamente imágenes de pizzas con las que se reentrenan (de ser posible) los modelos expuestos en la sección de estado del arte.

2.1.11. Métricas

A continuación se explicitarán las métricas utilizadas para realizar las comparaciones entre los modelos y sus formulas asociadas.

Es relevante notar que para poder calcular las métricas, primero, es necesario revisar a nivel de píxel las predicciones hechas por el modelo y compararlas píxel a píxel con las máscaras originales del conjunto de prueba. Con estas comparaciones se obtienen 4 posibles categorías:

- **Verdadero Positivo (TP)**: el modelo indica encontrar la clase buscada y efectivamente es la clase correcta.
- **Verdadero Negativo (TN)**: el modelo indica NO encontrar la clase buscada y efectivamente NO es la clase buscada.
- **Falso Positivo (FP)**: el modelo indica encontrar la clase buscada, pero NO es la clase correcta.
- **Falso Negativo (FN):** el modelo indica NO encontrar la clase buscada, pero SI es la clase objetivo.

Ahora que se tienen claros los conceptos de TP, TN, FP y FN, se procede a exponer las métricas utilizadas en función de estos valores:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{2.1}$$

$$Recall = \frac{TP}{TP + FN} \tag{2.2}$$

$$Precision = \frac{TP}{TP + FP} \tag{2.3}$$

$$Specifity = \frac{TN}{FP + TN} \tag{2.4}$$

$$F1 = \frac{2 * Precision * Recall}{Precision + Recall} \tag{2.5}$$

$$IoU : \frac{TP}{TP + FP + FN} \tag{2.6}$$

2.2. Estado del Arte

A continuación, el estado del arte tanto para segmentación semántica de objetos [4], como para la segmentación semántica para diferentes tipos de alimentos [3]. El primer algoritmo está enfocado en segmentar diferentes tipos de objetos teniendo un entrenamiento no supervisado, mientras que, el segundo está centrado en la capacidad de segmentar diferentes tipos de ingredientes dentro de platillos preparados, haciendo uso de un entrenamiento supervisado.

Sumado a los algoritmos anteriores [3][4], se expone DeepLabv3 [22]. Este es un modelo de aprendizaje supervisado del año 2017, el cual, está centrado en segmentación semántica mediante el uso de operaciones convolucionales dilatadas (*dilated convolutions*, también conocidas como *atrous convolutions*) lo que permite evitar la pérdida de información en redes profundas.

2.2.1. Estado del Arte de la Segmentación Semántica de Objetos

"*Self-supervised Transformer with Energy-based Graph Optimization*" o STEGO [4] por sus siglas en inglés, es un transformador auto supervisado creado para los desafíos de segmentación semántica de objetos *COCO-Stuff* [12] y *Cityscape* [14]. Donde "*COCO − Stuff*" o "*Common Objects in Context-Stuff*" es un popular desafío diseñado para impulsar el estado del arte en la segmentación semántica de las clases de "cosas", mientras que *Cityscape* es un desafió similar, pero enfocado en paisajes urbanos.

STEGO es capaz de descubrir y delimitar objetos conjuntamente sin supervisión humana, destilando características visuales no supervisadas, preentrenadas en grupos semánticos, usando una función de pérdida de contraste, mejorando significativamente lo que se venía desarrollando hasta la fecha [13] y acortando considerablemente la brecha existente respecto a sistemas de segmentación supervisada. Este segmentador funciona, a grandes rasgos, por cuatro bloques, los cuales se pueden observar en la figura 2.1. Es decir, un extractor de características "ViT", un segmentador, un bloque de "*cluster*" o agrupamiento y un CRF como salida. Cada uno de estos serán explicados a continuación.

En primera instancia, se tiene un extractor de características basado en *Vision Transformers* (ViT) llamado "DINO" [5] el cual puede producir delimitaciones (localizadas y de objetos sobresalientes) semánticamente relevantes. STEGO refina estas características obtenidas de su *backbone* y las usa para extraer información significativa entre las imágenes (semánticamente hablando). Es relevante destacar que, si bien para la creación de STEGO se centran en el extractor DINO debido a su calidad, STEGO puede funcionar con cualquier otro extractor de características.

Las características extraídas por el extractor, son entregadas a una red (*Segmentation Head*) "*feed forward*" con función de activación "ReLU", la cual tiene por finalidad aprender

una proyección no lineal en base al tensor de características entregado, formando grupos compactos y amplificando los patrones de correlación presente. Con el *backbone* y el cabezal de segmentación, se procede a crear la función de pérdida por destilación de correspondencia (*Correspondence Distillation Loss*), expuesta en la ecuación 2.7 (la cual es una simplificación de la función de pérdida real, de la cual se puede encontrar mayor información en [4]).

La función de pérdida L_{corr} es creada a partir de un par de imágenes, las cuales son procesadas previamente por el extractor de características y su salida es entregada a la red o cabezal de segmentación. Al procesar dichas imágenes se obtienen tensores para cada imagen, tanto del *backbone* como del cabezal, a los cuales se les aplica un producto tensorial (esta operación se aplica únicamente entre los tensores obtenidos por los mismos bloques, es decir, no se operan los tensores obtenidos por el *backbone* con los obtenidos por el cabezal de segmentación). Se define la nomenclatura para el producto tensorial del extractor de características como "F" pues, son las características de correspondencia o *Feature Correspondences*, por su nombre en inglés, mientras que al producto tensorial del cabeza de segmentación es defiendo como "S", debido a que son las correspondencias de segmentación o *Segmentation Correspondences*, por su nombre en inglés.

La finalidad de la función de pérdida es medir la distancia o diferencia entre los pares de tensores de características, proporcionalmente a sus correspondencias de características, lo cual y de forma simplificada, busca medir la diferencia o similitud entre pares de imágenes.

$$L_{corr} = -\sum(F - b)S \tag{2.7}$$

Además de F y S, se puede observar un hiperparámetro b, el cual agrega una "presión negativa" uniforme a la ecuación 2.7 para evitar que esta colapse. El minimizar L_{corr} con respecto a S fomenta que los elementos de S sean grandes, cuando los elementos de $F - b$ son positivos y pequeños cuando los elementos de $F - b$ son negativos. Esto debido a que los elementos de F y S son similitudes de coseno.

STEGO hace uso de esta función de pérdida L_{corr} en tres oportunidades para destilar características de correlación entre imágenes (es decir, similitudes entre las imágenes), la cuales son: entre la misma imagen, entre la imagen y sus K vecinos más cercanos y entre la imagen y una imagen aleatoria. Entre los K vecinos más cercanos la función de pérdida proporciona una atracción positiva, mientras que tiende a ser negativa entre la imagen y la imagen aleatoria. Lo anterior se puede ver ilustrado en la figura 2.1, la cual representa la arquitectura de STEGO y su proceso para el entrenamiento.

Finalmente, los componentes de la arquitectura faltantes son: el agrupamiento o *clustering* y el paso de refinamiento CRF [17]. Debido al proceso de obtención de características, las características de segmentación de STEGO tienden a formar grupos claramente delimitados. Se aplica un Algoritmo K-Means de minibatches, basado en la distancia del coseno [16], para extraer estos grupos y calcular asignaciones de clases concretas a partir de las características continuas de STEGO. Después de la agrupación, se refinan estas etiquetas con un CRF para mejorar aún más su resolución espacial.

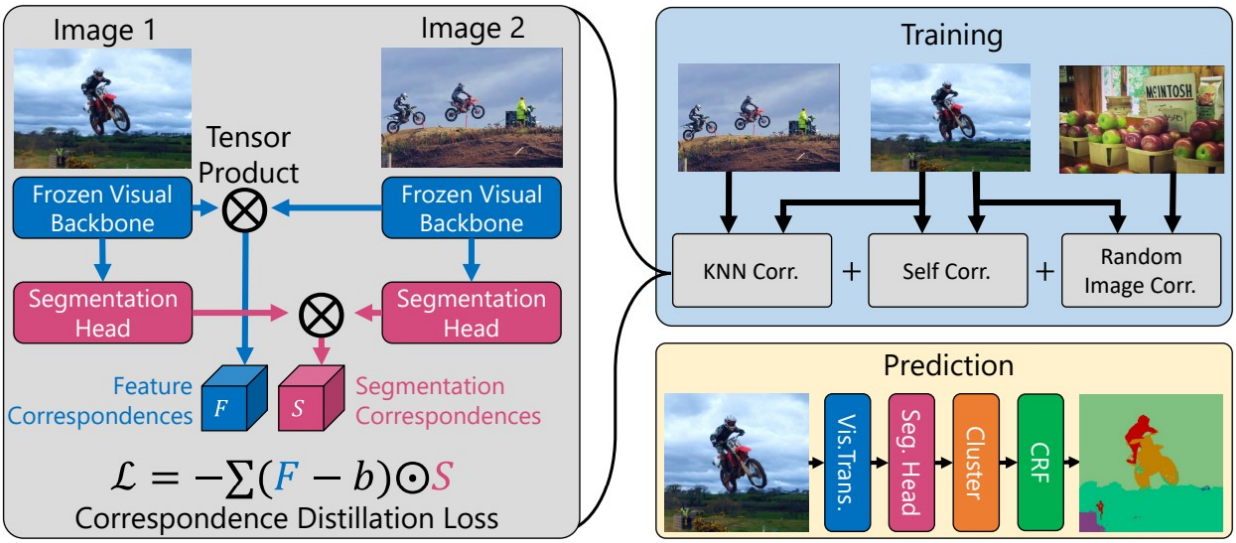


Figura 2.1: Descripción general de alto nivel de la arquitectura STEGO en los pasos de preparación y predicción [4].

2.2.2. Estado del Arte en Segmentación Semántica de Alimentos

Para la segmentación semántica de alimentos se utiliza [3]. En donde se presenta un enfoque de preentrenamiento multimodal llamado *ReLeM*, el cual se centra en la capacidad de segmentar diferentes tipos de ingredientes dentro de comidas o platillos preparados. Esta segmentación, en comparación con la segmentación semántica en imágenes de objetos generales, como la expuesta en STEGO [4], es más desafiante. Esto debido a la gran diversidad en la apariencia de los alimentos y la distribución a menudo desequilibrada de los ingredientes que componen los mismos.

El algoritmo presentado en [3] (que será llamado FoodSeg), es creado en base en aprendizaje supervisado, basando su aprendizaje en un conjunto de datos de entrenamiento previamente etiquetados. A grandes rasgos, cuenta con una estructura relativamente estándar para los problemas de segmentación, es decir, dos bloques consecutivos, uno de extracción de características seguido de un bloque de segmentación o clasificación.

El extractor de características utilizado por FoodSeg es "*Recipe Learning Module*" (ReLeM). Este es un método de preentrenamiento basado en multimodalidad, entrenado sobre datos de Recipe1M [18], dataset que cuenta con cerca de un millón de imágenes y recetas de cocina, las cuales contiene no solo el "cómo cocinar", sino, que, también contienen el "qué ingrediente usar", información utilizada sobre el modelo, llamando a esto transferencia de conocimiento multimodal. Específicamente, ReLeM integra datos de recetas de alimentos en formato de incrustación de lenguaje con la representación visual de imágenes de comida. De esta manera, se obliga a la representación visual de un ingrediente (que aparece en diferentes tipos de platos) a tener sus apariencias "conectadas" en el espacio de características. esto, a través de una incrustación de lenguaje común extraída de la etiqueta del ingrediente y sus instrucciones de cocción dadas en Recipe1M.

A continuación, se da paso al módulo de segmentación. Este módulo sigue el paradigma estándar de la segmentación semántica, donde la imagen de entrada se codifica primero en un codificador de visión (inicializado utilizando lo entrenado por ReLeM) y, luego, pasa por un decodificador de visión, el cual se inicializa aleatoriamente y se entrena para la segmentación de máscaras.

El proceso anteriormente descrito se puede observar en la figura 2.2, en donde se puede ver el flujo que sigue el algoritmo para realizar la segmentación deseada.

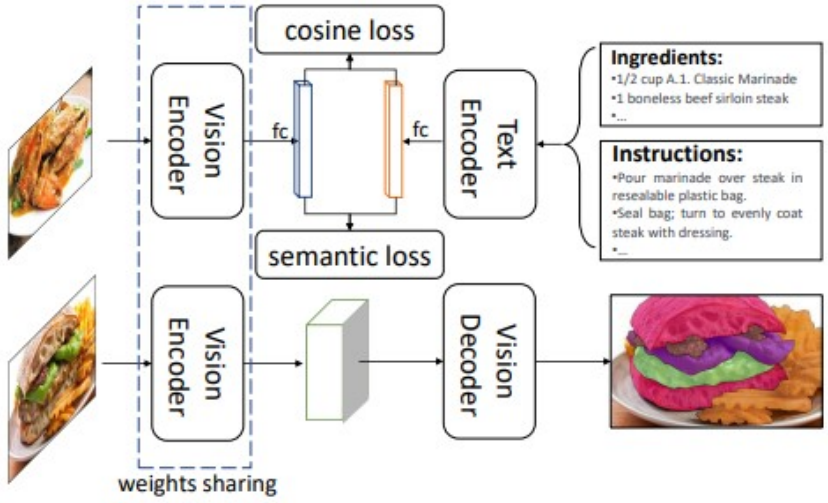


Figura 2.2: Estructura de "FoodSeg". Módulo de Aprendizaje de Recetas (ReLeM) y Módulo de Segmentación de Imágenes (Segmenter) [3].

Cabe destacar, que no existe un único modelo de segmentación, sino que, basándose en los diferentes diseños de codificador y decodificador, se pueden encontrar a tres grandes grupos: basados en dilatación, basados en *Feature Pyramid Networks* (FPN) y basados en $transformers$.

- **Basado en Dilatación:** las capas de convolución de dilatación tienen como objetivo agrandar la campos receptivos sin sacrificar la resolución
- **Basado en *Feature Pyramid Networks (FPN):*** integran mapas de características en diferentes capas al la conexión lateral. La representación de la imagen de capa superficial es mejorada al integrar los mapas de características generados en capas profundas.
- **Basado en *transformers:*** se basan en la atención [7]. Se adaptan bien a las tareas de segmentación semántica, en donde la información contextual es importante para segmentar objetos.

En la Figura 2.3, se exponen algunos resultados cualitativos del uso de un cabezal de segmentación semántica sin ReLeM (CCNet) y ReLeM haciendo uso de dicho cabezal (ReLeM-CCNet). Esto sobre el conjunto de prueba de FoodSed103, en donde se aprecia que ReLeM y el cabezal producen mejores predicciones que únicamente el cabezal de segmentación. Sin embargo, se puede observar en la última fila, que, al no tener una imagen con límites claros entre ingredientes, los resultados se ven afectados.

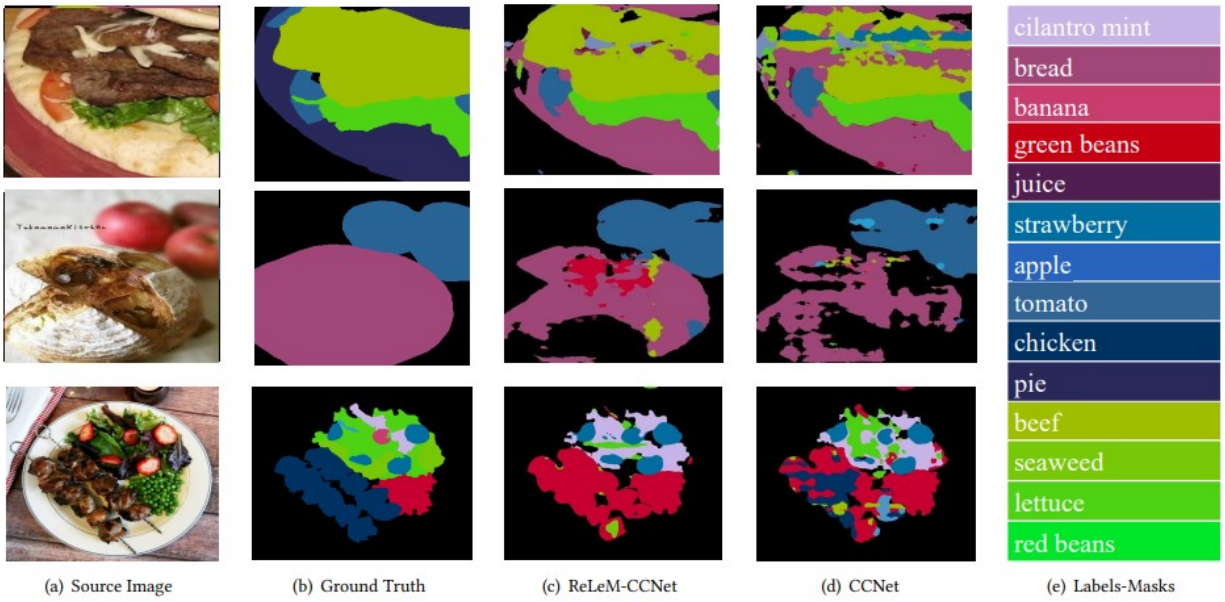


Figura 2.3: Resultados de visualización en FoodSeg103. Imágenes extraídas desde [3].

2.2.3. DeepLabv3 (Algoritmo Recomendado)

DeepLabv3 [22] está diseñado para la segmentación semántica de imágenes replanteando la forma del *kernel* en la operación de convolución para diferentes redes neuronales convolucionales profundas (DCNN, por sus siglas en inglés), llamando a este tipo de convoluciones: "convoluciones dilatadas" o "*atrous convolutions*". Al igual que FoodSed, este es un algoritmo de aprendizaje supervisado, es decir, para su correcto entrenamiento es necesario entregar información (imágenes o máscaras) etiquetada sobre los objetos que se buscan segmentar.

Si bien DeepLabv3 no es el estado del arte para la segmentación semántica (presentado durante el año 2017), este ha sido utilizado previamente por el equipo de trabajo de Kwali en desarrollos internos de la empresa. Debido a esto, dicho algoritmo es propuesto por la empresa para ser utilizado en la tarea de segmentación semántica.

El algoritmo utilizado como *backbone* o extractor de características corresponde a una red neuronal convolucional profunda (en este caso se utiliza una red ResNet-50 [23].Esta es una red neuronal convolucional con una profundidad de 50 capas que hace uso de información "residual" lo que ayuda a evitar la perdida de información), haciendo uso de convoluciones dilatadas. Dichas convoluciones son utilizadas en particular para superar el problema de reducción o pérdida de información causada por aplicar consecutivas operaciones de agrupamiento (*pooling*) o saltos de convolución (*convolution striding*) las cuales son utilizadas para aprender representaciones abstractas de los datos a costa de información detallada de los mismos. Es relevante notar que las convoluciones dilatadas (*atrous convolutions*) son una modificación de una operación de convolución normal, a las cuales se les agrega un nuevo parámetro: la tasa de dilatación. Esta tasa define el espacio entre los valores del kernel usado para la convolución. Así, un kernel de tamaño 3x3 con tasa de dilatación igual a 2 tendrá un campo de visión igual a un kernel de 5x5 de con tasa de dilatación 1, tal como se puede

observar en la figura 2.4.

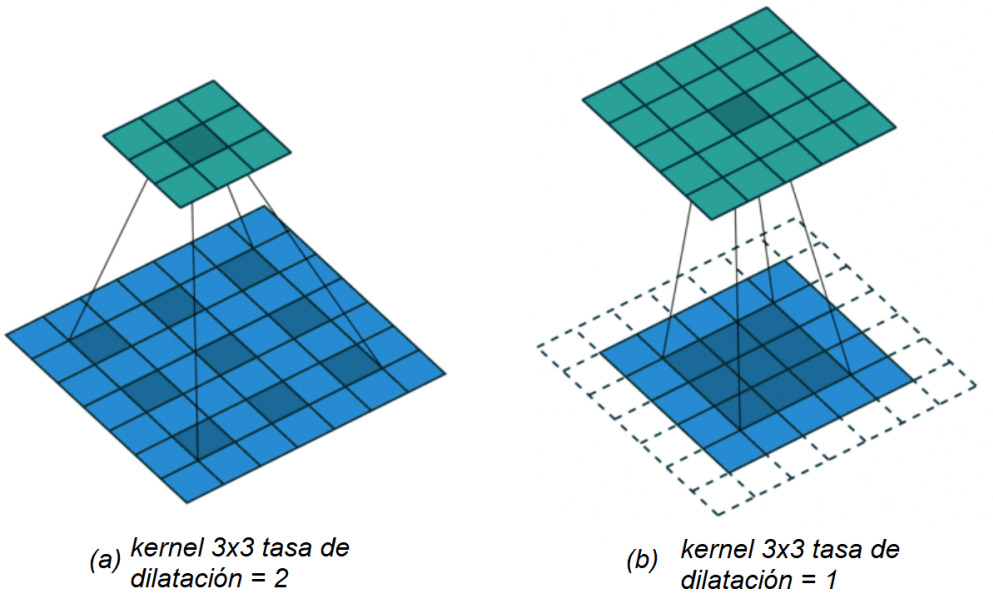


Figura 2.4: Kernel de 3x3 con diferentes tasas de dilatación.

Si bien, DeepLabv3 tiene una implementación más sencilla respecto, por ejemplo, a STEGO, al ser este un algoritmo supervisado implica que se necesitan datos previamente etiquetados para su entrenamiento. Luego, al ser los datos para esta misión muy particulares, no se tienen los necesarios, por lo tanto, para el reentrenamiento de DeepLabv3 es necesario crear dichos datos. También, cabe destacar que si bien se hace uso de una ResNet-50 como extractor de características, esta red no fue previamente entrenada para segmentar ingredientes de pizzas (sino, que, sobre datos de ImageNet, creado con cerca de 1.4 millones de imágenes), por lo tanto, se aplica un reentrenamiento sobre las últimas capas de la red, manteniendo intactas las demás capas.

Capítulo 3

Desarrollo

3.1. Validación Visual

Es relevante destacar por qué se deciden realizar validaciones visuales en primera instancia, en vez, de, por ejemplo, utilizar métricas con respaldo matemático (las cuales son utilizadas en el capítulo 4 de resultados).

Esto es, debido a que los estados del arte son modelos previamente entrenados sobre otros datos externos al problema, los cuales, no necesariamente tienen relación alguna con los datos utilizados por Kwali. Teniendo así, un modelo entrenado específicamente para alimentos (no ingredientes de pizzas) y otro no supervisado, del cual, se espera una gran capacidad de generalización.

Dado esto (y revisando las clases de los datos de los estados del arte) es posible afirmar que los modelos desconocen completamente (a nivel individual) los ingredientes típicos de pizzas (ya explicitados en la sección de objetivos). Con esto, medir la cantidad de verdaderos o falsos negativos, no tiene sentido, pues, el modelo desconoce las etiquetas de las clases buscadas, por lo tanto, realizar validación por métricas sin un previo reentrenamiento no es favorable para el proyecto.

Por otra parte, al reentrenar el algoritmo no supervisado, tampoco es necesario entregar clases de entrenamiento. Luego, para este algoritmo en particular, se deberían crear varios conjuntos de datos de validación (uno por cantidad diferente de ingredientes, los cuales son complejos de crear y aún más de obtener), en donde, realmente no se tendrían indicios previos de resultados correctos o mejoras.

Sumado a lo anterior, se cree que estos modelo entregan mayor información (y más amigable al observador) al observar las máscaras de predicción creadas. Pudiendo observar gráficamente, por ejemplo: como el modelo se intenta adaptar a ciertos ingredientes (lo que da indicios de poder dar buenos resultados de forma reentrenada), o por el contrario, si este confunde ingredientes entre clases o con objetos presentes en las imágenes, como, por ejemplo, la masa del borde de las pizzas con el color de las cajas de las mismas.

Por lo anteriormente explicado, se decide hacer en primera instancia una validación visual, la cual aporta una gran cantidad de información específica del problema, para así y en decisión de los resultados obtenidos por la validación visual, proceder con la validación matemática sobre los mejores modelos resultantes.

3.1.1. Validación Visual Sin Reentrenamiento

En la presente subsección se exponen las validaciones visuales de los algoritmos previamente expuestos, sin un reentrenamiento específico para el problema explicitado. Realizando la validación visual sobre datos reales de Kwali, además, de un breve análisis sobre estas. Cabe destacar que las imágenes utilizadas para la validación son escogidas de forma aleatoria sobre un conjunto de datos creado por Kwali, en donde se tienen 8 tipos de pizzas sin obstrucción, movimiento ni irregularidades respecto a la visibilidad de las pizzas. Simulando así, la entrada de una pizza promedio de Kwali al modelo.

3.1.1.1. Validación Visual de STEGO sin Reentrenamiento

Una vez explicada y profundizada la arquitectura de STEGO en las sección del estado del arte, se procede a implementar el segmentador para ser probado sobre los datos reales de Kwali, es decir, imágenes de pizzas tomadas desde desde la parte superior, tal como se puede observar en "Pizza Pepperoni" (imagen izquierda) de la figura 3.1.

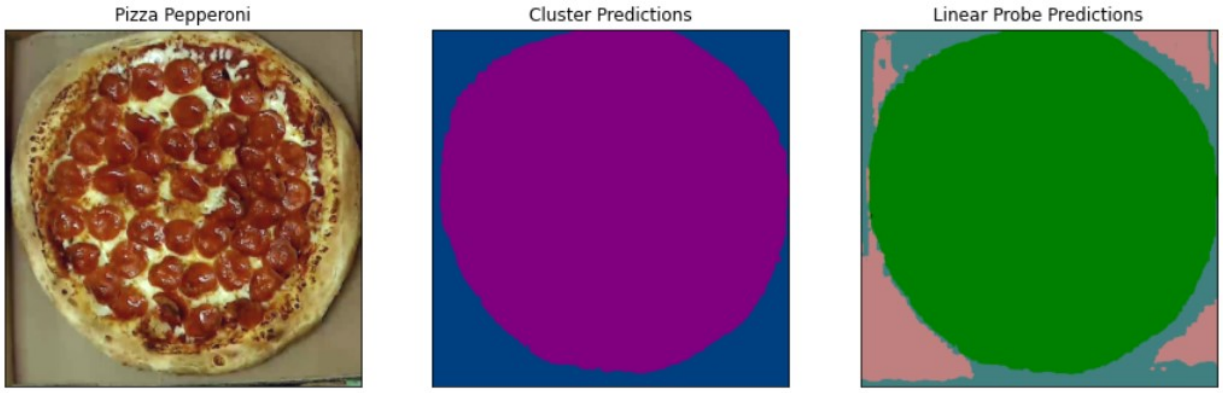


Figura 3.1: Ejemplo validación visual de STEGO sobre una pizza de pepperoni, imagen real de Kwali.

Como se puede observar en la figura 3.1, STEGO no da solución a la segmentación semántica de los diferentes ingredientes que se pueden encontrar en las variadas pizzas con las que trabaja Kwali, por lo tanto, sin un reentrenamiento, no se logra resolver el problema planteado.

El por qué STEGO no logra resolver el problema planteado (pese al hecho de que este presenta buenos resultados tanto en *COCO-Stuff* como en *Cityscapes*), es porque STEGO no ha sido específicamente entrenado para reconocer ingredientes de pizzas ni ingredientes de ningún otro tipo con los que trabaja Kwali. Por el contrario, sí fue entrenado para reconocer pizzas completas, pues, esto corresponde a una de las clases presentes en *COCO-Stuff*, segmentando de forma completa a las pizzas del fondo y no así, sus ingredientes, tal como se puede observar en la figura 3.1.

3.1.1.2. Validación Visual de FoodSeg sin Reentrenamiento

Si bien el algoritmo expuesto es un buen segmentador semántico de alimentos (respecto a los alimentos con los que fue entrenado), este no funciona de igual manera sobre clases o alimentos con los que no fue entrenado. Al revisar los ingredientes de foodseg103 presentes en el anexo de [3], se puede notar que, no todos los ingredientes típicos en las pizzas están incluidos en este dataset, además de que dicho conjunto de datos, incluye dentro de su categoría "*main*" la clase pizza como un solo objeto.

Así, se procede probar el algoritmo con diferentes modelos y combinaciones sobre imágenes reales de Kwali. Un ejemplo de esto, se puede observar en la figura 3.2. En la primera fila se exponen imágenes de una pizza (espinacas crema) sin haber sido procesadas, mientras que en la segunda fila se exponen las mismas imágenes, pero, procesadas con diferentes modelos entrenados sobre datos de FoodSeg103. En particular y haciendo referencia a la segunda fila: la primera columna es procesada con una FPN, mientras que la imagen de la segunda columna es procesada con ReLeM y la misma FPN anterior. Por otra parte, la tercera imagen es procesada con CCNET y, finalmente, la última imagen es procesada con ReLeM y la CCNET anterior (notar que tanto CCNET como ReLeM-CCNET son expuestos en la figura 2.3 sobre ingredientes para los que si fueron entrenados).

Se puede notar en la figura 3.2, específicamente en (a), que la FPN no es capaz de detectar ningún ingrediente interno de la pizza, sin embargo, detecta gran cantidad de la pizza en si, exceptuando ciertos trozos. Además, intenta segmentar parte del borde de la misma. Observando (b) es posible notar que al agregar ReLeM a la FPN se logra detectar más área de la pizza, pero nuevamente no se logra segmentar ningún ingrediente. Caso similar a los anteriores ocurre con CCNET, la cual delimita casi en su totalidad a la pizza, como un unico ingrediente, dejando una fracción mucho más reducida sin detectar, esta vez, casi sin detectar partes del borde. Finalmente, se tiene ReLeM-CCNET la cual es capaz de detectar toda la pizza, pero, sin segmentar ninguno de los ingredientes, ni tampoco secciones del borde. Sin embargo, esta ya no deja secciones de la pizza sin delimitar. Cabe notar que todos los modelos fueron capaces de detectar en parte el ají que se encuentra en la esquina inferior izquierda de la imagen, ingrediente que es detectado como parte de la pizza.

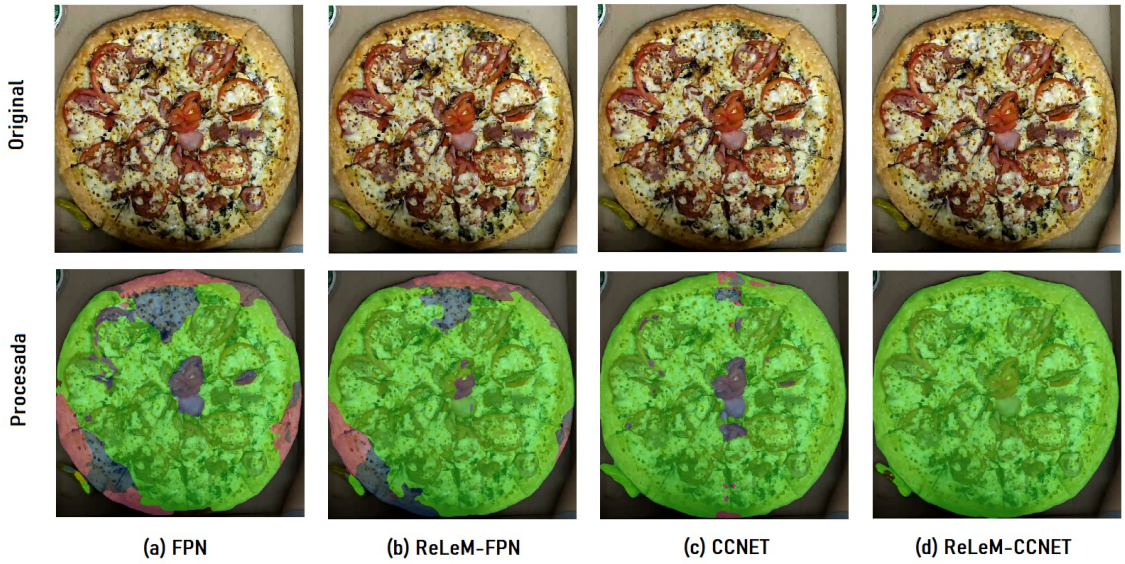


Figura 3.2: Resultados de visualización en FoodSeg103 sobre pizza "espinacas crema". Imagen extraídas del dataset de Kwali.

Si bien, se cree que FoodSeg mejoraría su capacidad para delimitar los diferentes ingredientes en las pizzas al ser reentrenada, para lograr dicho entrenamiento específico (considerando que para lograr los resultados expuestos en la figura 2.2 utilizaron cerca de 1 millón de datos etiquetados), se cree se necesitan demasiados datos etiquetados semánticamente (por lo menos un 1 % de los datos de entrenamiento), lo cual, no es factible de obtener por los medios actuales. Por lo anterior, no se podrá realizar el reentrenamiento al algoritmo para el problema en particular.

3.1.1.3. Validación Visual de DeepLabv3 sin Reentrenamiento

Como se mencionó en la sección del estado del arte, subsección DeepLabv3, este algoritmo no corresponde al estado del arte para segmentación semántica, pero, al haber sido utilizado por los miembros de Kwali con anterioridad, hizo que este fuera recomendado por la empresa. En particular, este algoritmo de segmentación, fue utilizado por la empresa para segmentar pizzas del resto de la imagen, similar al resultado de la predicción del *cluster* en la imagen central de la figura 3.1, obteniendo resultados como los expuestos en la figura 3.3. Dicha figura representa una imagen genérica de Kwali en el lado izquierdo, mientras que en el lado derecho se expone la máscara sobrepuesta a la imagen original, donde, se ve claramente que segmenta a la pizza completa del fondo, pero, ningún ingrediente en particular. Esto dado su entrenamiento específico, por lo tanto, no resuelve el problema planteado. Debido a esto, se procede a realizar su reentrenamiento sobre los datos específicos.

Figura 3.3: Pizza de pepperoni segmentada del fondo. Imagen original de Kwali.

3.1.2. Validación Visual con Reentrenamiento

3.1.2.1. Datos de Entrenamiento

A continuación, en la figura 3.4 se exponen ejemplos representativos de las imágenes de cada uno de los tipos de pizzas utilizadas para realizar los reentrenamientos, tanto de STEGO como de DeepLabv3. Es relevante notar que pese a que utilizaron la misma clase de datos (haciendo referencia a los tipos de pizzas), estos modelos son entrenados con diferentes cantidades de estos mismos datos (utilizando 200 imágenes para las pruebas bajo 5000 max_steps y 400 para las pruebas con mayor cantidad, a menos que se explicite lo contrario, esto dado por limites en las máquinas donde se realizaron dichas pruebas Google Colaboratory y Amazon Web Services), ya sea respecto a cantidad de imágenes por clase de pizza o por cantidad total de imágenes de pizzas.

Figura 3.4: Ejemplos representativos de las imágenes utilizadas para realizar los reentrenamientos. Imágenes de Kwali.

En los 8 tipos de pizzas expuestas en la figura 3.4, se pueden encontrar, a grandes rasgos, los siguientes ingredientes por tipo de pizza, en donde, es relevante notar que, dependiendo la franquicia de origen de la pizza (entre mismos tipos de pizzas), podrían existir ingredientes diferentes. Además, de poder ser retirados por petición de los consumidores finales de las mismas.

- **bbq_chicken_bacon:** salsa barbacoa (bbq), pollo, jamón (tocino) y cebolla.
- **garden_fresh:** tomate, aceitunas, vegetales verdes (pimentón verde), champiñones y cebolla.
- **meatball_pep:** albóndigas y pepperoni.
- **meats:** carne roja, pepperoni y jamón.
- **pepperoni:** pepperoni.
- **philly_cheesesteak:** carne roja y vegetales verdes (pimentón verde).
- **spinach_tomato_alfredo:** tomate, vegetales verdes (espinaca), champiñones y salsa Alfredo.
- **works:** pepperoni, jamón, carne roja, cebolla, aceituna y vegetales verdes (pimentón verde).
- **Todas:** queso, masa (corteza) y algunas también sectores con salsa de tomate.

Teniendo así un total de, aproximadamente, 14 ingredientes: salsa barbacoa (bbq), pollo, cebolla, tomate, aceituna, vegetales verde, champiñones, albóndigas, pepperoni, carne roja, jamón, salsa de tomate, queso y corteza (no se considera la salsa Alfredo debido a que esta es indistinguible del queso, pese a esto, le da una coloración más blanca que se debe tener en cuenta para DeepLabv3). Estos, distribuidos en las 8 clases de pizzas expuestas. Además, para STEGO, se agregan algunas pizzas con la categoría *Unkown* para realizar pruebas, las cuales corresponden a pizzas no identificadas por los algoritmos de Kwali (esto por la forma, tamaño o distribución irregular de dicha pizza) pese a ser pizzas pertenecientes a alguno de los 8 tipos expuestos, como por ejemplo, una pizza de pepperoni con una forma particular de distribuir los pepperonis, tal como se observa en la imagen de la figura 3.5.

Figura 3.5: Ejemplos imagen desconocida o irregular, imagen real de Kwali.

3.1.2.2. STEGO Reentrenado con Datos de Kwali

Se realizó una revisión exhaustiva que tardó alrededor de 3 meses para la correcta implementación y correcto reentrenamiento de STEGO sobre los datos específicos. Pese a esto, dichos esfuerzos no dieron los frutos esperados al realizar la validación visualmente, contrastando la imagen entregada contra la imagen de salida.

A continuación, se exponen los resultados más relevantes obtenidos por STEGO sobre los datos específicos, en donde, si bien se mantuvo la estructura original del algoritmo, se probaron diferentes hiper parámetros. En particular, se hicieron pruebas sobre la cantidad máxima de pasos de entrenamiento ("max_step" una forma indirecta de modificar las épocas de entrenamiento), número de clases y tamaño del *batch*. Aunque para realizar diferentes pruebas también se modificaron, por ejemplo, el número de datos, la clase de datos (utilizando un único tipo de pizza), número de vecinos más cercanos, entre otros.

En un principio se tenía previsto encontrar alrededor de 14 clases de ingredientes. Esto considerando diferentes ingredientes típicos en pizzas. Sin embargo, al realizar los primeros entrenamientos y revisar visualmente los resultados, fue posible percatarse de que el algoritmo intenta encontrar el número total de clases en cada imagen, independiente de la cantidad real de ingredientes en ella. Tal como se puede observar en las imágenes de la figura A.1 presente en el anexo. Por lo anterior, se decidió reducir el número de clases por encontrar, como se puede apreciar en la figura A.2 también presente en el anexo, bajando los ingredientes buscados desde 14 hasta 6.

Al notar una mejora en los resultados debido a la baja de clases buscadas de 14 a 6, se decide volver a realizar pruebas, aumentando el número de max_step desde 100 hasta 500 y 5000, entrenando el modelo para 10 y 4 clases diferentes, respectivamente, con lo cual se obtienen los resultados expuestos en la figura A.3 del anexo, en donde se puede apreciar la mejora al disminuir la clases buscadas y aumentar la cantidad máxima de pasos de entrenamiento. Con lo anterior, se puede observar una significativa mejora en la segmentación al contrastar los resultados obtenidos en la figura A.1 con los resultados obtenidos en la figura A.4.

Aunque al aumentar el número de máximo de pasos de entrenamiento y disminuir la cantidad de clases por buscar, los resultados mejoraron, al pasar de 5000 a 15000 pasos máximos y duplicar los datos de entrenamiento (llegando a las 400 imágenes), la mejora disminuyó, haciendo que la salida de la red empeore respecto al experimento con 5000 pasos máximos, perdiendo la segmentación más fina que se estaba logrando con anterioridad, lo que se puede apreciar de mejor forma al comparar entre las pizzas de *pepperoni* y la clasificada como *unknown* entre las figuras A.4 y A.5, ambas presentes en el anexo.

Finalmente, al percatarse de la baja en los resultados obtenidos, se procede a realizar un nuevo (y se adelante, último) entrenamiento sobre STEGO, esta vez de 30000 pasos máximos, modificando además la tasa de aprendizaje del modelo, disminuyéndola desde $5e-4$ hasta $5e-6$, esperando mejorar los resultados del modelo, debido a los resultados poco satisfactorios obtenidos hasta el momento. Con esto y después de un entrenamiento considerablemente más largo que los demás (14 horas contra las 4 horas del entrenamiento anterior), se obtienen los resultados expuestos en la figura 3.6, en donde, si bien se obtienen mejoras considerables respecto los ingredientes más finos en la predicción del cluster, aún se presentan fallas significativas en la predicción linear (salida de la red). Además, y pese a que el algoritmo mostró mejoras respecto a la predicción del *cluster*, este sigue presentando resultados no satisfactorios.

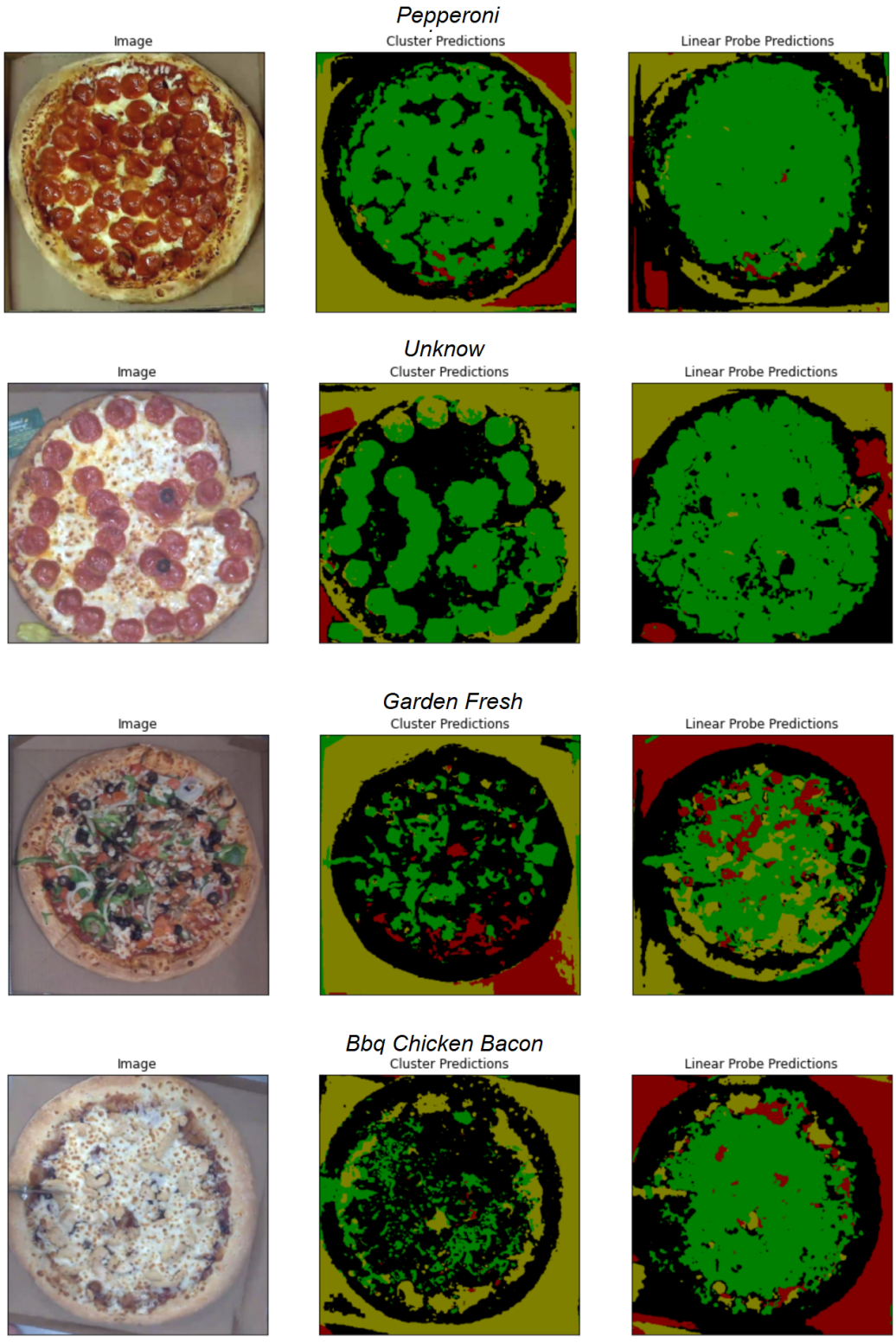


Figura 3.6: Visualización de resultados de STEGO sobre pizzas: pepperoni, unknown, garden fresh y bbq chicken bacon. 4 clases por pizza, 30000 max_step con 400 imágenes balanceadas de entrenamiento.

Debido a los resultados expuestos (los cuales se cree no se podrán mejorar mucho más

con los medios actuales), el tiempo invertido, sumado además a la falta de control sobre la detección de clases (pues el algoritmo siempre intenta segmentar según la cantidad de clases con las que fue entrenado, más no así, según la cantidad real de clases presentes en la imagen de entrada) y la necesidad de entrenamientos cada vez más largos (los cuales dejaron de realizarse de forma gratuita en Google Colaboratory debido a su falta de capacidad, haciendo uso de una máquina pagada en Amazon Web Services), incentivaron a dejar de reentrenar STEGO, pues, finalmente, se considera que no podrá resolver adecuadamente el problema planteado dados los resultados obtenidos.

3.1.2.3. DeepLabv3 Reentrenado con Datos de Kwali

A diferencia de STEGO, este algoritmo es supervisado, por lo tanto, es necesario realizar un *dataset* etiquetado, tanto de entrenamiento y validación, como uno de prueba (tarea realizada en SuperAnnotate [24]). Si bien, originalmente se creía necesario tener al menos algunos cientos de datos etiquetados por ingrediente para el entrenamiento de DeepLabv3 de forma correcta, al tener apenas 23 imágenes etiquetadas, se pudo realizar un pequeño entrenamiento de prueba, de las cuales se obtuvieron 18 de las 23 imágenes etiquetadas por una empresa externa a Kwali, a donde se mandó a etiquetar un pequeño conjunto de datos específicos, para segmentar el tipo de queso de las pizzas, intentando segmentar por cantidades dadas por diferentes criterios, ya sean dados por los clientes o por criterios propios de Kwali. Este pequeño conjunto de datos mandados a etiquetar, corresponden a una prueba realizada para ver si es que una guía de etiquetado de tipos de quesos hecha por el autor del documento era lo suficientemente robusta para que anotadores externos pudieran entender y segmentar manualmente en SuperAnnotate de forma correcta. Por lo tanto, no son las mejores máscaras creadas, pero, si lo suficientemente buenas como para ser utilizadas a modo de prueba. Si bien, se tienen criterios específicos para considerar diferentes cantidades de queso, para temas de este documento, se trabaja principalmente en segmentar el ingrediente queso correcto, correctamente, por sobre los ingredientes bajo y exceso de queso.

A continuación, se puede observar en la figura 3.7 cómo son las máscaras creadas, en donde si bien se presentó una pizza con la máscara queso correcto, también se tienen máscaras para cantidades de bajo y exceso de queso.

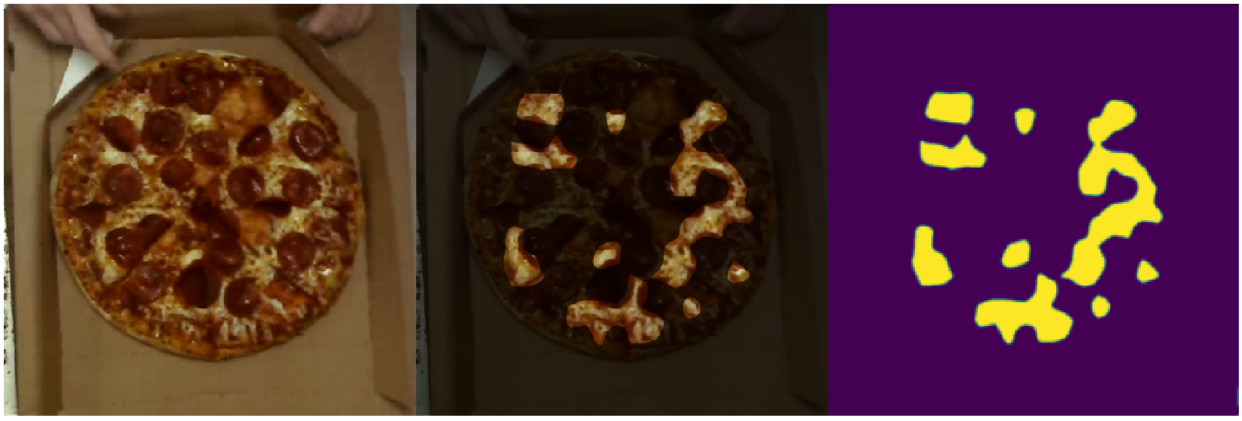
(a) pepperoni_56.jpg (b) máscara e imagen juntos (c) máscara de pepperoni_56.jpg

Figura 3.7: Máscara de queso correcto, pizza de pepperoni, imagen original de Kwali.

Con estas diferentes cantidades de queso, se entrenaron 2 algoritmos DeepLabv3 de prueba, uno especializado en queso correcto y otro especializado en exceso de queso. En donde se considera exceso de queso al queso que casi no presenta o derechamente no tiene puntos cafés (creados por la cocción misma del queso), tiene generalmente un color más blanco que el resto y no tiene coloraciones rojas dadas por la salsa de tomate (lo que indicaría bajo queso). Es relevante notar que dependiendo del tipo de pizza, el queso puede tener una coloración más o menos blanca, como se puede ver parar el caso de las pizzas tipo *philly_cheesesteak* y *spinach_tomato_alfredo* expuestas en la figura 3.4. Por lo tanto, para realizar una correcta segmentación en los datos de entrenamiento se tiene que tener este punto y algunos otros puntos similares en cuenta al momento de realizar la creación de los datos de entrenamiento, es decir, agregar sesgo al modelo dado por información especifica entregada por Kwali.

Inicialmente, se entrenó con una época para probar el correcto funcionamiento del código (es decir, que el código se ejecute sin advertir errores). Al intentar mostrar los resultados, el segmentador de queso no segmentó nada de la imagen. Debido a esto, se procedió a reentrenar al algoritmo, esta vez con 50 épocas, en donde este consiguió algunas segmentaciones relativamente aceptables (considerando la baja cantidad de datos) sobre el ingrediente correcto, lo que contrastado contra el mismo segmentador y contra los estados del arte, se consideran mejoras e indicios de que el algoritmo sirve para la problemática descrita.

Motivado por esta pequeña mejora, se procede a entrenar un segmentador de exceso de queso, con 150 épocas, obteniendo los resultados expuestos en las imágenes de la figura 3.8, y de la figura B.1 presente en el anexo del documento, en donde se expone: (a) la imagen original. (b) la máscara sobre puesta y (c) solo la máscara resultante, sin la imagen de la pizza.



(a) Original (b) Segmentación (c) Máscara

Figura 3.8: DeepLabv3 entrenado con 23 imágenes y 150 épocas para segmentar el exceso de queso, prueba sobre pizza de pepperoni. Imagen de kwali.

Aunque los resultados expuestos obtenidos en las pruebas no son perfectos (pues, los datos de entrenamiento no son del todo correctos), sí son bastante acertados para haber sido entrenados con una cantidad tan baja de datos (apenas 23 imágenes y sus máscaras, en donde 19 fueron utilizadas para el entrenamiento y 4 para la validación, esto realizado internamente por DeepLabv3). Motivados por estos buenos resultados, se decidió crear los datos necesarios en SuperAnnotate para entrenar segmentadores DeepLabv3 especializados en cada ingrediente.

Se realizaron pruebas al entrenar DeepLabv3 para segmentar tanto, bordes de la pizza, como, para segmentar los pepperonis de las pizzas. Los datos utilizados corresponden a 25 imágenes de pizzas con borde y con pepperoni, creando máscaras como las expuestas en las imágenes de la figura 3.9.

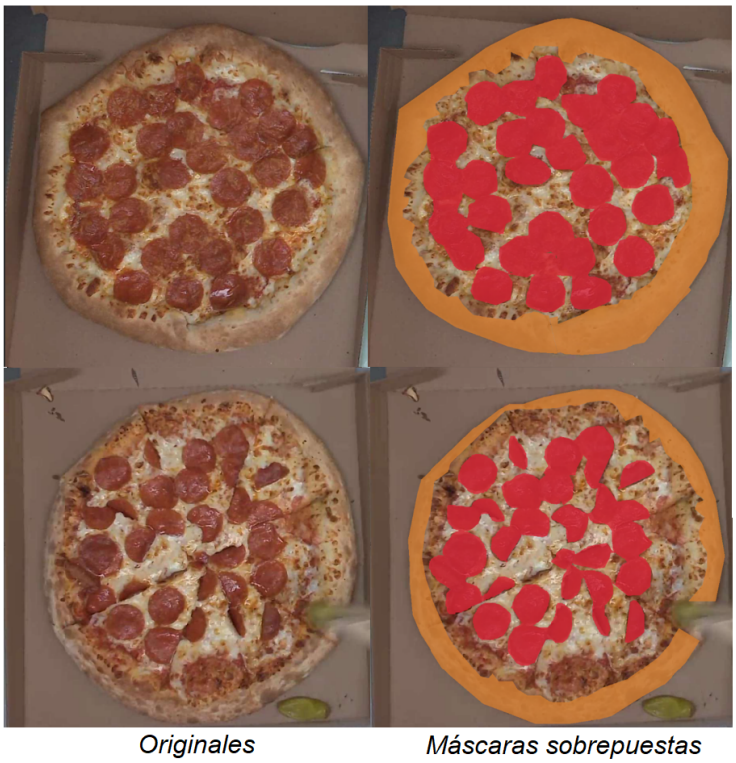


Figura 3.9: Imágenes con máscaras de pepperoni y borde de la pizza sobre puestos.

Así, al repetir los entrenamientos de 150 épocas, ahora con 25 datos (20 de entrenamiento y 5 de validación de forma interna en DeepLabv3) se obtuvieron los resultados expuestos en las figuras 3.10 y 3.11

(a) Original (b) Segmentación (c) Máscara creada

Figura 3.10: Resultados de DeepLabv3 con entrenamiento de 150 épocas, segmentación del masa (borde) sobre pizzas: *pepperoni*, *meats* y *philly_cheesesteak*.

Figura 3.11: Resultados de DeepLabv3 con entrenamiento de 150 épocas, segmentación de pepperonis sobre pizzas: *pepperoni*, *meats* y *meatball_pepperoni*.

Así, analizando los resultados obtenidos por los 2 modelos DeepLabv3 reentrenados, es posible observar, sobre todo en las imágenes de la figura 3.11, que, si bien el segmentador delimita un gran cantidad de los pepperonis presentes, este segmenta también gran parte de otros ingredientes como pepperonis, como el caso de la segunda y tercera fila de la figura mencionada, en donde, se tiene una pizza *meats* con pepperoni (los cuales a simple vista son difíciles de distinguir, pues parecen salsa de tomate) y una pizza *meatball_pepperoni*. En donde se segmentan como pepperoni todas las albóndigas puestas sobre o cerca de los pepperonis, clase para la cual fue entrenado el modelo.

Para solucionar esto, se cree que puede ser útil ampliar el uso del algoritmo para que sea capaz de identificar más que un solo tipo de ingrediente por pizza, con lo cual, se espera que el algoritmo aprenda de mejor forma los diferentes límites entre los ingredientes. Sin embargo, también se cree que aumentando la cantidad de datos de entrenamiento, épocas

o modificando la tasa de aprendizaje del modelo, se obtendrán mejores resultados que los expuestos en la figura 3.11 (entrenamientos realizados con una tasa de aprendizaje de 1e-4 y un *batch* de tamaño 4, mismos con los que se entrenó el modelo original de Kwali para la segmentación del fondo de las pizzas). Por lo tanto, se procede a realizar la creación de un *dataset* de mayor tamaño, esto, para 11 diferentes clases de ingredientes y así continuar con el reentrenamiento de dos diferentes versiones de modelos de DeepLabv3. Una versión especializada, en un único ingrediente y otra versión multiclase, en donde, para obtener los resultados sobre los mismos ingredientes, se crearon 11 modelos especializados, cada uno en un ingrediente diferente. Buscando ambos tipos de modelos distinguir los mismos 11 ingredientes: pepperoni, albóndigas, pollo, carne roja, vegetales verdes (pimentón verde y espinaca), aceitunas, tomate, masa (borde) y queso dividido en 3 categorías (exceso, correcto y bajo). Para cada uno de los 8 tipos de pizzas expuestos en la figura 3.4, se tiene la siguiente distribución de ingredientes que se buscan segmentar según el tipo de pizza:

- **bbq_chicken_bacon:** pollo.
- **garden_fresh:** tomate, aceitunas, vegetales verdes (pimentón verde).
- **meatball_pep:** albóndigas y pepperoni.
- **meats:** pepperoni, carne roja y jamón juntos como etiqueta carne.
- **pepperoni:** pepperoni.
- **philly_cheesesteak:** carne roja y vegetales verdes (pimentón verde).
- **spinach_tomato_alfredo:** tomate, vegetales verdes (espinaca).
- **works:** pepperoni, carne roja y jamón juntos como carne, aceituna y vegetales verdes (pimentón verde).
- **Todas:** bajo queso, queso correcto y exceso de queso, además de la masa (corteza o borde).

Así, se procede a segmentar un conjunto de 59 imágenes especialmente sobre las categorías de quesos (de las cuales se tiene al menos la presencia de una de las 3 categorías: bajo, correcto y exceso) en cada una de las pizzas con las que trabaja Kwali. Con este nuevo *dataset* se procede a reentrenar modelos especializados en 2 de los 3 tipos de quesos (exceso y correcto), para continuar con la validación visual y observar posibles mejoras en los resultados al aumentar los datos. Las máscaras de estas segmentaciones se pueden observar en las imágenes (b) de las figuras 3.12 y 3.13,

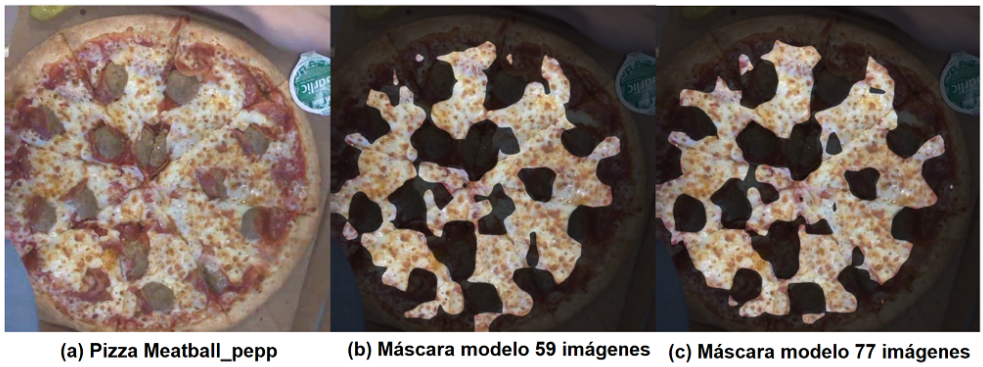


Figura 3.12: Resultados de DeepLabv3 con entrenamientos de 600 épocas, batch = 4, tasa de aprendizaje = 1e-4. 59 y 77 imágenes cada modelo. Segmentación de queso correcto sobre pizzas *meatball_pepperoni*.

Figura 3.13: Resultados de DeepLabv3 con entrenamientos de 600 épocas. 59 y 77 imágenes cada modelo. Segmentación de exceso de queso sobre pizzas *meatball_pepperoni*.

Así, realizando una ultima validación visual, es posible notar que al aumentar los datos desde 59 a 77 imágenes, las máscaras resultantes son más ajustadas al ingrediente real que se busca segmentar, además, de aprender de mejor forma cuando el ingrediente no se encuentra presente en la imagen. Tal como se puede apreciar en las imágenes (b) y (c) de la figura 3.13. Esta última, presenta una pizza de pepperoni sin exceso de queso, sin embargo el modelo entrenado con 59 imágenes detecta erróneamente un sector con exceso de queso, mientras que el modelo entrenado con 77 imágenes es capaz de "comprender" que no hay presencia de este exceso de queso. Es relevante notar que si bien el modelo de 77 imágenes no segmenta exceso de queso en esta ocasión, este sigue presentando pequeñas fallas al segmentar.

Así, con los resultados de la segmentación obtenidos, se procede a ampliar el conjunto de datos y realizar pruebas para encontrar el o los mejores segmentadores, los cuales son evaluados con diferentes métricas, de lo cual, se profundiza con mayor detalle en el capítulo 4, de resultados, dando por finalizada así la validación visual. Escogiendo a DeepLabv3 como mejor opción de los 3 algoritmos expuestos.

Capítulo 4

Resultados

Como se pudo observar en el capitulo anterior, de los 3 modelos expuestos, solo DeepLabv3 pasó la validación visual y es prometedor para realizar una correcta segmentación semántica sobre los ingredientes explicitados. Por esto, y dado que ya se realizaron algunas pruebas sobre un dataset de 77 imágenes (del cual se concluyó se obtendrán mejores resultados al aumentar los datos y variar métricas), se procedió a aumentar la cantidad de datos para entrenar el modelo, de lo cual se profundiza a continuación:

4.1. Datos de Entrenamiento y Prueba

Para realizar los nuevos entrenamientos y posteriores evaluaciones (sobre un conjunto de prueba), se aumentaron los datos de entrenamiento y validación desde 77 imágenes a 97 imágenes. Además, se creó un pequeño conjunto de prueba de 18 imágenes, el cual, esta compuesto por 2 imágenes de cada uno de los 8 tipos de pizza expuestos en la figura 3.4. Sumado a eso, se agregaron 2 imágenes extras, una del tipo *pepperoni* y otra del tipo *meatball_pepperoni*.

Sobre el dataset de 97 imágenes, se sigue buscando segmentar los mismo 11 ingredientes anteriormente explicitados. Esto, con la misma distribución (respecto a que ingrediente va sobre que tipo de pizza), teniendo la siguiente distribución según de pizza:

- **bbq_chicken_bacon:** x14
- **garden_fresh:** x12
- **meatball_pep:** x6
- **spinach_tomato_alfredo:** x13
- **pepperoni:** x25
- **philly_cheesesteak:** x9
- **works:** x7
- **meats:** x11

Como no se busca clasificar el tipo de la pizza, sino, que, los tipos de ingredientes, es relevante conocer la cantidad de imágenes que tienen "X" ingrediente, esto para el conjunto de entrenamiento y el conjunto de prueba, tal como se puede apreciar en la tabla 4.1. En donde el conjunto de validación se separa del conjunto de entrenamiento, hecho internamente por DeepLabv3, seleccionando 19 imágenes aleatoriamente.

Tabla 4.1: Distribución de los ingredientes en los datos de entrenamiento y prueba. Respecto a cuantas imágenes contienen al menos una muestra (píxel) clasificado como X ingrediente.

INGREDIENTE	**ENTRENAMIENTO**	**PRUEBA**
Pepperoni	48	10
Meatball	6	3
Meat	23	5
Green_Veg	40	8
Olive	19	4
Chicken	16	2
Crust	97	18
Right_Cheese	97	18
Low_Cheese	68	8
Excess_Cheese	74	13
Tomato	24	4

4.2. Resultados Modelo DeepLabv3

A continuación, se exponen los mejores resultados obtenidos en la búsqueda del mejor modelo DeepLabv3 para resolver la problemática planteada.

En la tabla 4.2 se pueden observar los resultados de la búsqueda para la mejor taza de aprendizaje, manteniendo fijo el tamaño del *batch* en 4. Esto, para el modelo multiclase de DeepLabv3, mientras que en la tabla 4.3 se busca el mejor *batch*, para la mejor tasa de aprendizaje encontrada. Cabe destacar que se decide fijar el *batch* en tamaño 4, esto debido a que, originalmente, el modelo de DeepLabv3 usado por Kwali, utilizaba específicamente este tamaño de batch

Tabla 4.2: Resultados búsqueda mejor tasa de aprendizaje, batch fijo en 4. DeepLabv3 multiclase.

LR	ACC	RECALL	PRECISION	SPECIFITY	F1	mIoU
1e-3	0.9702	0.6017	0.5957	0.9861	0.5944	0.4449
1e-4	**0.9715**	**0.5867**	**0.6096**	**0.9859**	**0.5941**	**0.4453**
2e-5	0.9718	0.5277	0.6065	0.9859	0.5585	0.4155
1e-6	0.2050	0.9754	0.1067	0.1774	0.1535	0.1058
1e-7	0.1216	0.9832	0.0783	0.0783	0.1232	0.0782

Como se puede apreciar en la tabla 4.2, los mejores resultados se obtienen con la tasa de aprendizaje de 1e-4, esto tomando en cuenta tanto la métrica *precision*, como el promedio de la intersección sobre la unión ($mIoU$) del modelo. Luego, con la mejor tasa encontrada ($lr = 1e-4$), se procede a realizar experimentos para encontrar el mejor *batch* para resolver el problema de segmentación expuesto.

Tabla 4.3: Resultados de la búsqueda del mejor batch, manteniendo el lr fijo en 1e-4, modelos DeepLabv3 multiclase.

BATCH	ACC	RECALL	PRECISION	SPECIFITY	F1	mIoU
2	0.9718	0.5740	0.6115	0.9860	0.5893	0.4419
4	0.9715	0.5867	0.6096	0.9859	0.5941	0.4453
6	**0.9708**	**0.6099**	**0.6008**	**0.9855**	**0.6011**	**0.4509**
8	0.9725	0.5794	0.6246	0.9857	0.5984	0.4487
10	0.972	0.5853	0.6133	0.9854	0.5952	0.4462
12	0.972	0.5745	0.6102	0.9858	0.5875	0.4386
16	0.9713	0.5808	0.6005	0.9855	0.5864	0.4380

Una vez encontrada la mejor combinación respecto al tamaño del batch y la tasa de aprendizaje, se decide entrenar 11 modelos únicos especializados con los mismos datos y parámetros. Esto, para poder contrastar entre ambos modelos. Los resultados obtenidos por los modelos especializados en un solo ingrediente se exponen en la tabla 4.4. Es relevante notar que se utilizan 200 épocas de entrenamiento para cada uno de los modelos especializados en una clase, mientras que se utilizaron 400 épocas para los modelos multiclase, esto debido a que se comienza a sobre entrenar en ambos casos antes de llegar a los valores dichos.

Dado que se tienen 11 modelos diferentes, se decide tomar 3 representativos para graficar el *total_loss* obtenido durante los entrenamientos. Para la elección de los modelos, se decidió tomar los modelos de queso correcto, pepperoni y pollo. Esto dado, pues, los 2 primeros además de ser junto con *crust* los ingredientes más repetidos en el conjunto de datos, también son el primer y tercer mejor segmentador respecto a $mIoU$. Mientras que para la elección del tercer modelo, simplemente se escogió uno que no se encuentre entre los mejores o peores resultados, tomando indistintamente al modelo de pollo (*Chicken*) para graficar su *total_loss*, obteniendo los resultados expuestos en la gráfica de la figura 4.1.

Tabla 4.4: Resultados 11 modelos especializados en 1 clase cada uno, lr fijo en 1e-4, batch fijo en tamaño 6.

Ingrediente	ACC	RECALL	PRECISION	SPECIFITY	F1	mIoU
Excess	0.97	0.474	0.394	0.982	0.43	0.274
Right	0.891	0.884	0.803	0.895	0.842	0.726
Low	0.964	0.383	0.401	0.982	0.392	0.244
Meat	0.988	0.343	0.343	0.994	0.343	0.207
Olive	0.996	0.672	0.55	0.997	0.605	0.434
Chicken	0.992	0.547	0.499	0.996	0.522	0.353
Meatball	0.943	0.84	0.15	0.95	0.254	0.204
Green_Veg	0.987	0.843	0.519	0.989	0.642	0.474
Tomato	0.986	0.689	0.545	0.991	0.609	0.438
Crust	0.947	0.881	0.793	0.959	0.835	0.716
Pepp	0.971	0.831	0.794	0.983	0.812	0.683
PROM	**0.967**	**0.672**	**0.526**	**0.974**	**0.571**	**0.432**

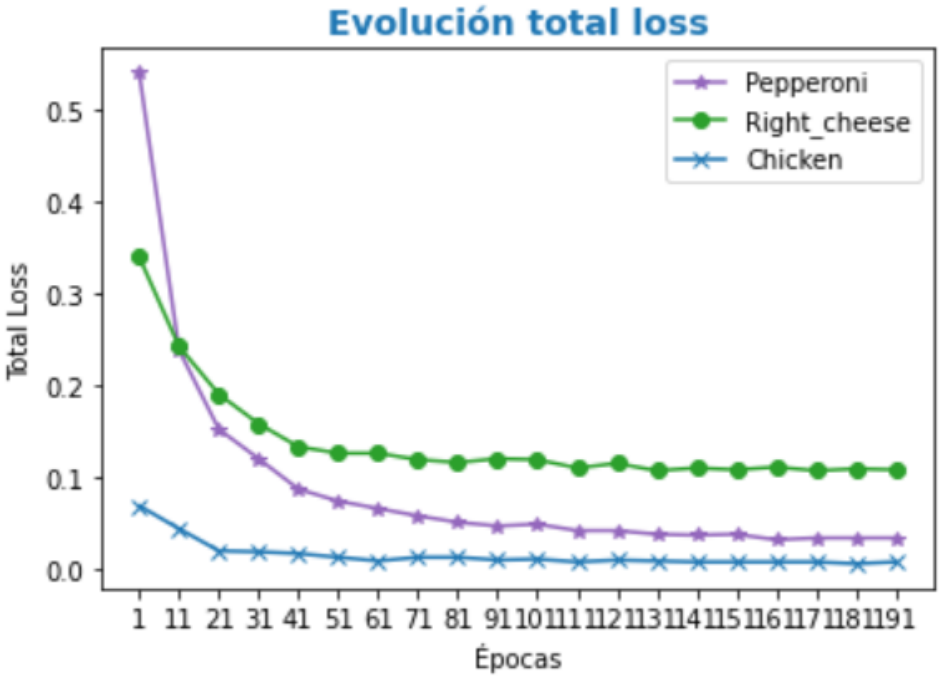


Figura 4.1: Total_loss modelos: *pepperoni*, *right_cheese* y *chicken*. Entrenados con 200 épocas. Batch_size = 6 y lr = $1e-4$

Ahora que fue descubierto el mejor modelo DeepLabv3 multiclase (con batch 6 y lr 1e-4), se procede a comparar sus resultados con los obtenidos por los 11 segmentadores de 1 clase cada uno. Esto se puede observar al revisar la tabla 4.5, en donde se exponen consecutivamente las clases correspondientes del los 11 modelos con las clases respectivas del modelo unificado.

Tabla 4.5: Comparación resultados mejor modelo unificado ("modelo_M", lr=1e-4 y batch=6) vs 11 modelos especializados en 1 clase cada uno ("modelo_1", lr = 1e-4 y batch = 6).

Ingrediente	ACC	RECALL	PRECISION	SPECIFITY	F1	mIoU
Excess_1	0.97	0.474	0.394	0.982	0.43	0.274
Excess_M	0.966	0.483	0.350	0.978	0.406	0.255
Right_1	0.891	0.884	0.803	0.895	0.842	0.726
Right_M	0.881	0.739	0.877	0.950	0.802	0.669
Low_1	0.964	0.383	0.401	0.982	0.392	0.244
Low_M	0.968	0.437	0.473	0.985	0.454	0.293
Meat_1	0.988	0.343	0.343	0.994	0.343	0.207
Meat_M	0.986	0.392	0.319	0.992	0.352	0.213
Olive_1	0.996	0.672	0.55	0.997	0.605	0.434
Olive_M	0.994	0.566	0.462	0.997	0.509	0.341
Chicken_1	0.992	0.547	0.499	0.996	0.522	0.353
Chicken_M	0.993	0.42	0.551	0.990	0.477	0.313
Meatball_1	0.943	0.84	0.15	0.95	0.254	0.204
Meatball_M	0.994	0.71	0.775	0.998	0.741	0.589
Green_Veg_1	0.987	0.843	0.519	0.989	0.642	0.474
Green_Veg_M	0.988	0.734	0.568	0.992	0.640	0.471
Tomato_1	0.986	0.689	0.545	0.991	0.609	0.438
Tomato_M	0.986	0.587	0.569	0.993	0.578	0.407
Crust_1	0.947	0.881	0.793	0.959	0.835	0.716
Crust_M	0.950	0.836	0.836	0.971	0.836	0.719
Pepp_1	0.971	0.831	0.794	0.983	0.812	0.683
Pepp_M	0.973	0.805	0.829	0.987	0.817	0.690
PROM_1	**0.967**	**0.672**	**0.526**	**0.974**	**0.571**	**0.432**
PROM_M	**0.972**	**0.610**	**0.601**	**0.986**	**0.601**	**0.451**

Capítulo 5

Análisis Final

Por el contexto en el cual se desarrolló la memoria, se considera que es más relevante poder medir la correcta o incorrecta clasificación de la clase objetivo, es decir, los verdaderos y falsos positivos, más que los verdaderos y falsos negativos. En particular, los verdaderos negativos, otorgan muy poca información, esto, por la baja cantidad de clases objetivo en varias de las imágenes, pues se hace una máscara por clase, en donde la gran mayoría de los píxeles de las imágenes no pertenecen al ingrediente buscado. De igual forma, muchas veces el ingrediente directamente no se encuentra en la imagen evaluada.

Por lo explicado, se decidió poner especial atención tanto en la métrica de $mIoU$ como en $F1$ y $Precision$, pues, estas centran sus evaluaciones en los TP. Sin embargo, también es relevante tener en cuenta las malas detecciones, es decir, los píxeles correspondientes a la clase objetivo que el modelo no logró reconocer (FN). Por esto, el orden de importancia dado a las métricas expuestas es el siguiente:

Tabla 5.1: Orden de relevancia de las métricas.

	ACC	RECALL	PRECISION	SPECIFITY	F1	mIoU
Lugar	5º	4º	3º	6º	2º	1º

De igual forma, es importante notar por qué no se centra la decisión final en el *accuracy* o en $specifity$. Esto, pues, dichas métricas hacen uso de los verdaderos negativos (TN) en sus numeradores. Luego, como se tienen solo imágenes desbalanceadas (respecto a la cantidad de píxeles de una u otra clase), con muy pocos píxeles por detectar como clase objetivo, la detección de verdaderos negativos siempre es mucho más elevada respecto al resto de posibilidades (TP, FP, FN).

Haciendo uso de las métricas descritas y en el orden explicitado, es posible notar al observar la tabla 4.2, que, el mejor modelo DeepLabv3 multiclase corresponde al modelo con una tasa de aprendizaje de $1e-4$. Mientras que observando la tabla 4.3, se puede afirmar que el modelo multiclase que mejor resuelve el problema planteado, es el modelo DeepLabv3 con $batch = 6$ y una tasa de aprendizaje de $lr = 1e - 4$.

Dado los resultados expuesto en la sección de validación visual, es posible afirmar que DeepLabv3 reentrenado tiene un mejor desempeño para la tarea de segmentar semánticamente los diferentes ingredientes presentes en las pizzas con las que trabaja Kwali. Esto en contraste con ambos estados del arte para segmentación semántica y el propio modelo DeepLabv3 sin reentrenamiento (considerando a DeepLabv3 no reentrenado, como, el modelo de segmentación de fondo de pizzas, el cual, ya manejaba Kwali). Es decir, para la tarea especificada, DeepLabv3 superó el desempeño de STEGO (con y sin reentrenamiento) y FoodSeg sin reentrenamiento. Como se mencionó con anterioridad, debido al limitante por la falta de datos debidamente etiquetados, no fue posible realizar un correcto reentrenamiento de FoodSeg sobre los datos expuestos.

Ahora, si bien se tiene que DeepLabv3 reentrenado, es el mejor modelo para resolver la tarea planteada (esto respecto STEGO, FoodSeg y el mismo DeepLabv3 sin reentrenar), es necesario contrastar las 2 versiones de DeepLabv3, es decir, el modelo multiclase contra los 11 modelos especializados.

Si solo fuera relevante el IoU total del modelo, sería sencillo declarar a DeepLabv3 multiclase como el mejor modelo. Sin embargo, si se considerara más relevante, por ejemplo, una clase en particular, se podría considerar a DeepLabv3 especializado, como el mejor modelo (en esa clase particular). Ahora, considerando que el trabajo realizado tiene como finalidad acoplarse al flujo de trabajo de Kwali, hay otros factores que hay que tener en consideración, como, por ejemplo, el costo extra asociado (tanto económico como computacional) de tener que mantener 11 modelos funcionando al mismo tiempo en AWS, esto, en vez de un único modelo que realiza lo mismo. Sumado a lo anterior, hay que considerar la complejidad de manejar los modelos separadamente, lo que podría provocar a largo plazo una mayor cantidad de errores asociados únicamente al manejo de los modelos.

Si bien un modelo unificado tiene ventajas, el uso de varios modelos especializados también las tiene, como, por ejemplo, el hecho de que para mejorar el desempeño en una clase (o inclusive para crear más datos de entrenamiento para esa clase) solo es necesario centrarse en dicha clase, dejando intactas a las demás. Siendo así, mucho más rápido un reajuste en caso de ser necesario. Sin embargo, para Kwali se considera más valioso tanto el resultado de las métricas, como el costo de implementación del modelo y la facilidad de uso. Esto por sobre la sencillez para realizar nuevos entrenamientos o creación de nuevos datos.

Así, con lo anteriormente expuesto y en vista de los resultados de las métricas seleccionadas, considerando el costo extra asociado (computacional y económicamente hablando) y la facilidad de uso, se considera a DeepLabv3 multiclase (con una tasa de aprendizaje de $1e-4$ y un batch de tamaño 6), la mejor elección de modelo para resolver el problema de segmentación semántica para alimentos específicos (ingredientes) que se encuentran en las pizzas con las que trabaja Kwali.

Capítulo 6

Conclusiones

El aporte más relevante que se logró alcanzar con la realización de la presente memoria, es la implementación y reentrenamiento de un segmentador semántico especializado (DeepLabv3), capaz de identificar específicamente los ingredientes (explicitados con anterioridad en la sección "3.1.2.3. DeepLabv3 reentrenado con datos de Kwali") más relevantes presentes en las pizzas de diferentes franquicias con las que trabaja Kwali. Cumpliendo así, el objetivo general de la memoria. Además, se espera que en un futuro próximo y haciendo uso de otro algoritmo (con la salida del algoritmo expuesto como entrada), se pueda medir, en tiempo real, la cantidad en masa asociada a los ingredientes en la preparación de cada una de las pizzas y, por ende, sus costos monetarios. Información sumamente relevante tanto para Kwali como para sus clientes.

Otra contribución relevante que se realizó al llevar a cabo el proyecto, fue la creación de un *dataset* segmentado semánticamente, compuesto por 115 imágenes. Las cuales tienen 11 diferentes tipos de ingredientes (clases) y 8 diferentes tipos de pizzas. Estos conjuntos de datos fueron necesarios para realizar el entrenamiento, la validación y pruebas, sobre los modelos entrenados con el segmentador especializado, explicitado con anterioridad. Es relevante notar que dicho *dataset* dista mucho del volumen de datos del utilizado para entrenar a FoodSeg (cerca de 1 millón de datos) y por lo tanto, no sirve para un correcto reentrenamiento de dicho algoritmo.

Sumado al algoritmo final presentado y al conjunto de datos creados, se aportó con la creación y mejoramiento de una guía de quesos especializada, para poder mandar a etiquetar grandes conjuntos de imágenes de pizzas, segmentando el queso en estas, en las categorías de: correcto, bajo y exceso. Con esto, se espera que Kwali pueda hacerse con un gran conjunto de datos semánticamente etiquetados para futuros trabajos. Desafortunadamente, por temas de confidencialidad no es posible exponer dicha guía.

Además, es relevante notar que si bien, se expone a DeepLabv3 multiclase con los resultados de la tabla 4.3 como el mejor modelo, se cree, que, los modelos aún tiene un amplio margen de mejora. Para lograr esto, se cree necesario, únicamente, aumentar los datos de entrenamiento. Hay que tener en cuenta que al aumentar los datos es posible necesitar realizar nuevas pruebas para encontrar el mejor tamaño de *batch* y el mejor valor para la tasa de aprendizaje, mas no se haya necesario cambiar de algoritmo.

Finalmente, y no menos importante, se aportó con una investigación, implementación, comparación y análisis de los estados del arte actuales y el algoritmo previamente utilizado por Kwali, además de justificar el por qué los actuales estados del arte para la segmentación semántica no son capaces de resolver la problemática planteada.

Bibliografía

[1] P. I. Orellana Rueda “Segmentación Semántica y reconocimiento de lugares usando características CNN preentrenadas”.

[2] William A. Castrillon, Damian A. Alvarez, Andrés F. López. “Técnicas de extracción de características en imágenes para el reconocimiento de expresiones faciales”.

[3] Xiongwei Wu, Xin Fu, Ying Liu, Ee-Peng Lim, Steven C.H. Hoi, Qianru Sun. “A Large-Scale Benchmark for Food Image Segmentation”

[4] M. Hamilton, Z. Zhang, B. Hariharan, N. Snavely, W. T. Freeman “Unsupervised Semantic Segmentation by Distilling Feature Correspondences”.

[5] M. Caron, H. Touvron, I. Misra, H. Jegou, J. Mairal, P. Bojanowski, and A. Joulin. “Emerging properties in self-supervised vision transformers”.

[6] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, N. Houlsby. “An image is worth 16x16 words: Transformers for image recognition at scale”.

[7] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser. “Attention Is All You Need”.

[8] C. A. Ruiz, M. S. Basualdo, D. J. Matich. “Redes Neuronales: Conceptos Básicos y Aplicaciones”.

[9] K. O’Shea and R. Nash. “An Introduction to Convolutional Neural Networks”.

[10] https://healthdataminer.com/data-mining/aprendizaje-supervisado-y-no-supervisado/

[11] https://itelligent.es/es/category/procesamiento-del-lenguaje-natural/

[12] H. Caesar, J. Uijlings, and V. Ferrari. “COCO-Stuff: Thing and Stuff Classes in Context”

[13] J. H. Cho, U. Mall, K. Bala, and B. Hariharan. “Picie: Unsupervised semantic segmentation using invariance and equivariance in clustering”.

[14] https://www.cityscapes-dataset.com/

[15] A. Palmer, J.J. Montaño, R. Jiménez. “Tutorial sobre Redes Neuronales Artificiales: El Perceptrón Multicapa”.

[16] James MacQueen. “Some methods for classification and analysis of multivariate observations”.

[17] J. Lafferty, A. McCallum, F. Pereira. “Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data”.

[18] A. Salvador, N. Hynes, Y. Aytar, J. Marin, F. Ofli, I. Weber, and A. Torralba. “Learning cross-modal embeddings for cooking recipes and food images”.

[19] https://www.kwali.ai/how-it-works-1

[20] V. Vargas Baeza. “Sistema de visión artificial para el control de calidad de piezas cromadas”. Tesis, 2020.

[21] Xuan Bi, Xiwei Tang, Yubai Yuan, Yanqing Zhang, and Annie Qu. “Tensors in Statistics” Annual Review of Statistics and Its Application, 2021.

[22] Liang-Chieh Chen, George Papandreou, Florian Schroff, Hartwig Adam. “Rethinking Atrous Convolution for Semantic Image Segmentation”.

[23] Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun. “Deep Residual Learning for Image Recognition”. arXiv:1512.03385, 2015.

[24] https://www.superannotate.com/

Anexos

Anexo A. Resultados Visuales, Reentrenamiento de STEGO

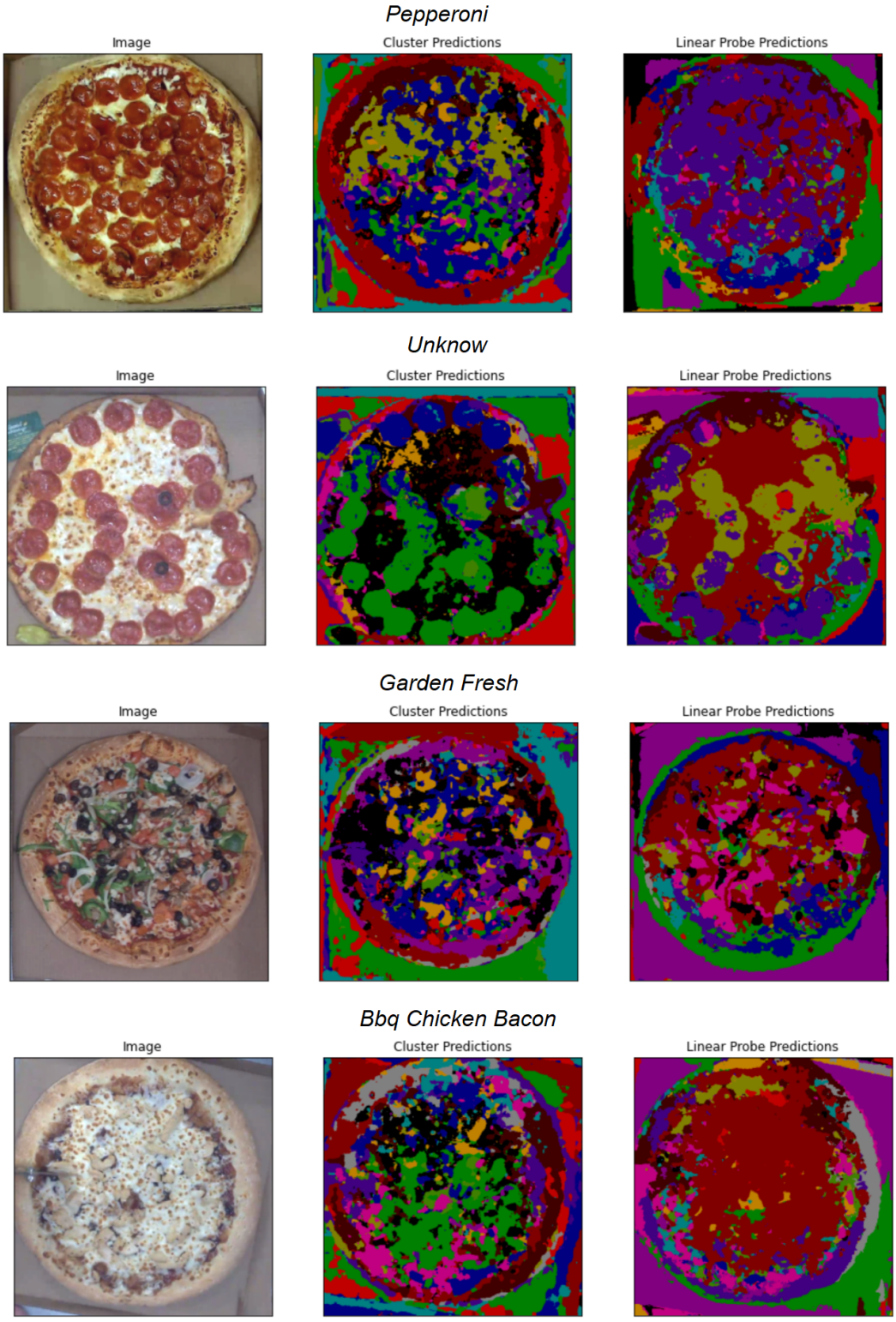


Figura A.1: Visualización de resultados de STEGO sobre pizzas pepperoni, unknown, garden fresh y bbq chicken bacon, 14 clases, 100 max_step con 200 imágenes de entrenamiento.

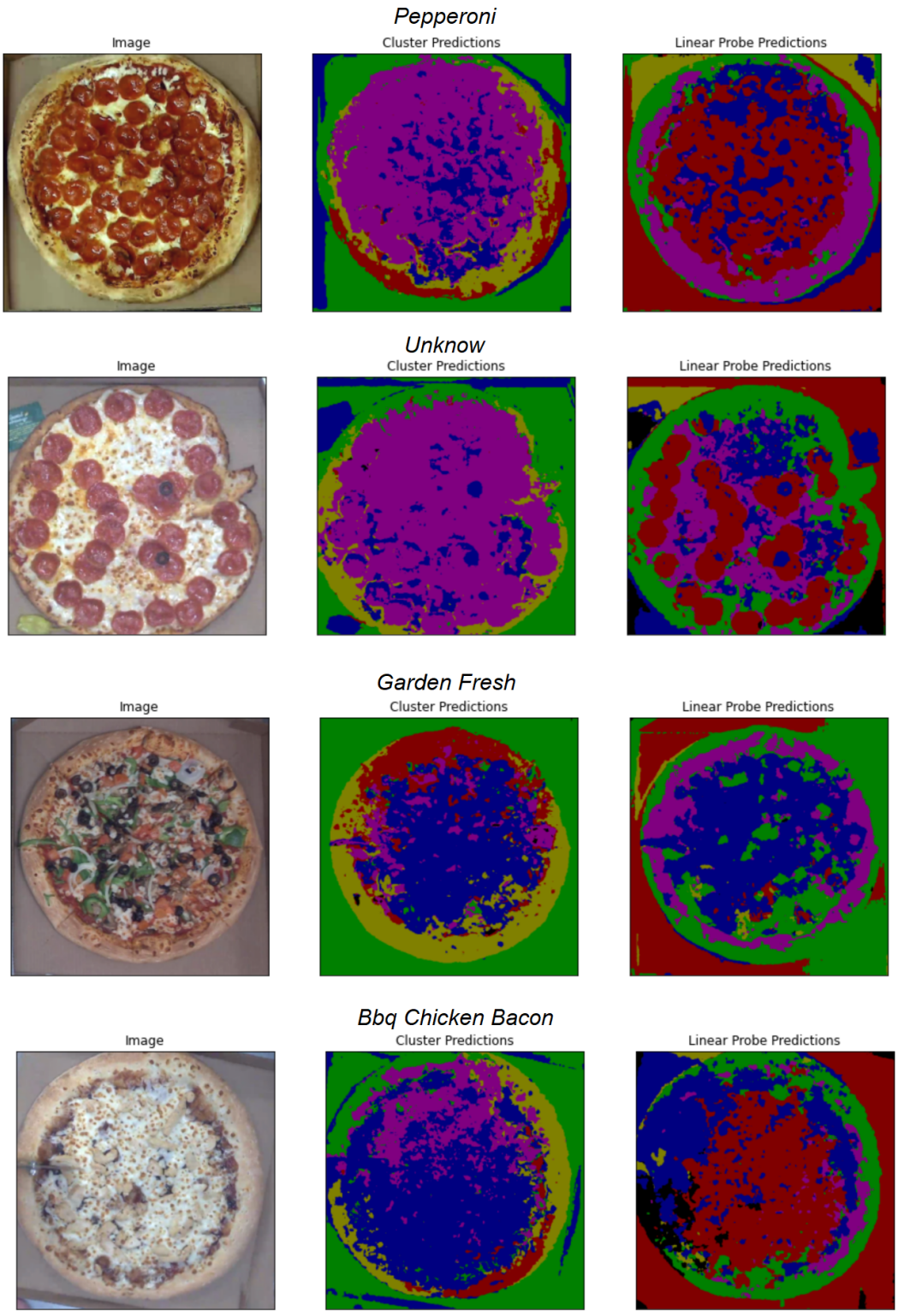


Figura A.2: Visualización de resultados de STEGO sobre pizzas pepperoni, unknown, garden fresh y bbq chicken bacon, 6 clases, 100 max_step con 200 imágenes de entrenamiento.

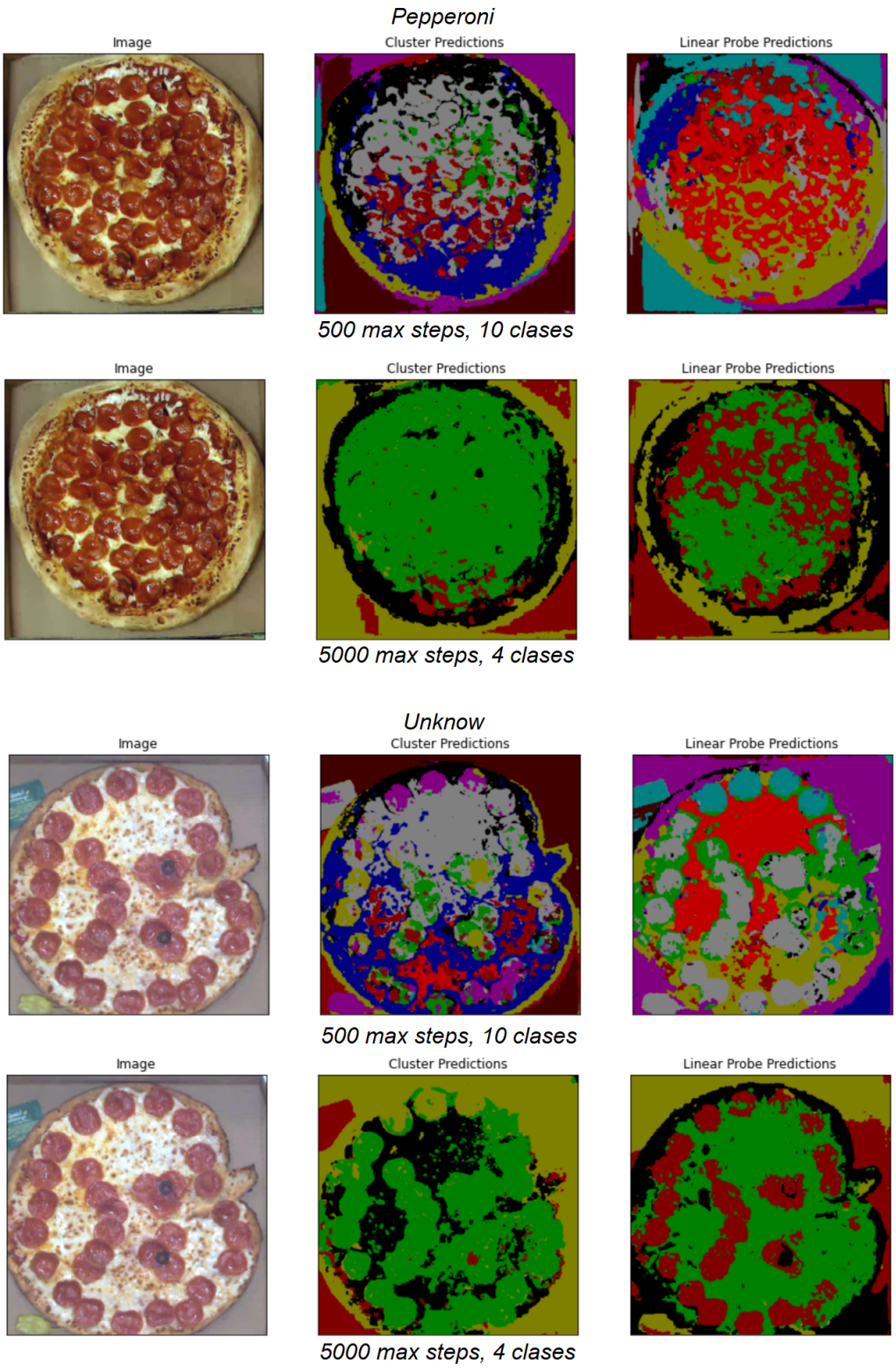


Figura A.3: Visualización de resultados de STEGO contrastados sobre pizzas pepperoni y unknown, para 4 clases y 5000 max steps contra 10 clases y 500 max steps, con 200 imágenes de entrenamiento.

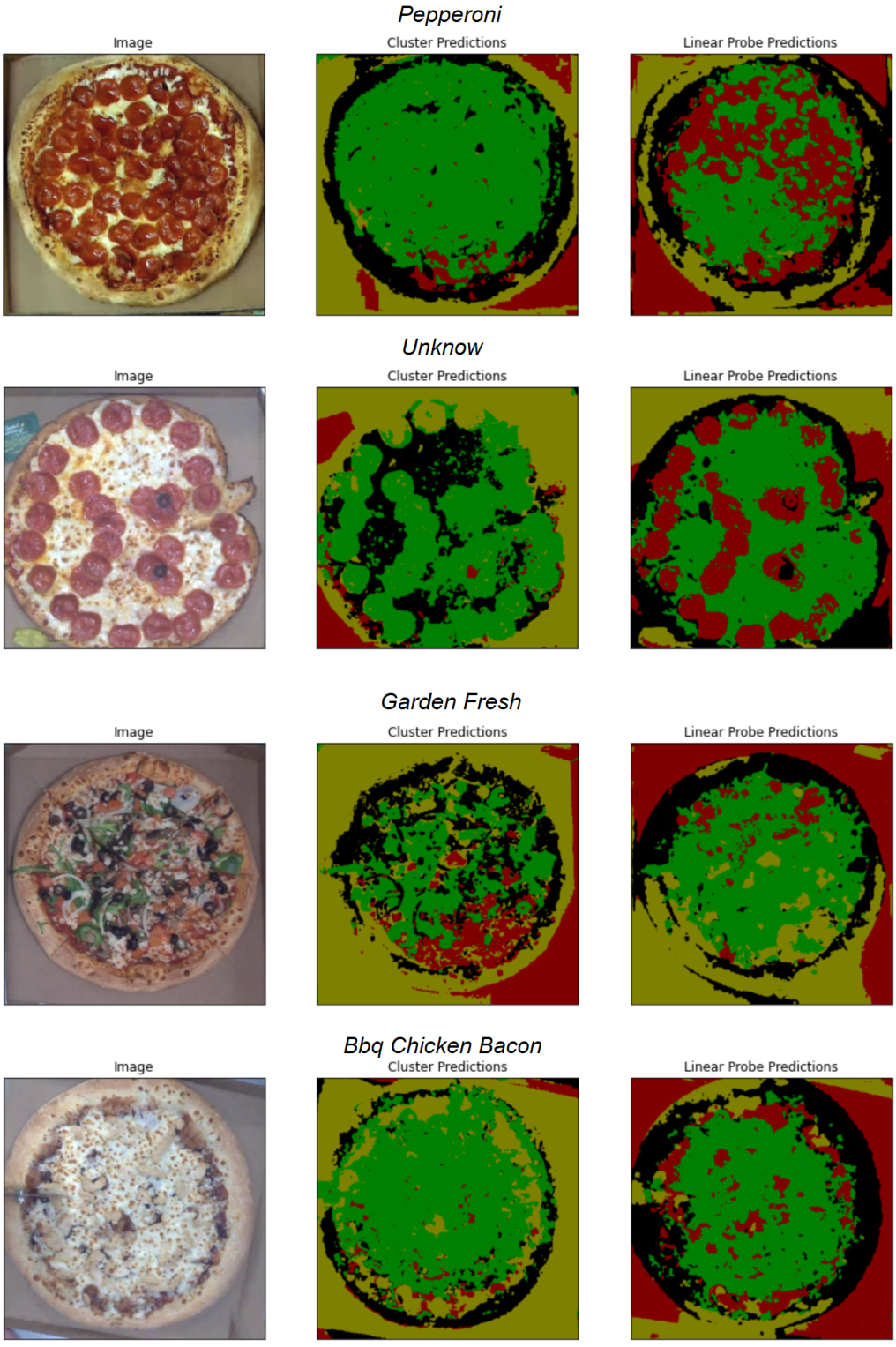


Figura A.4: Visualización de resultados de STEGO sobre pizzas pepperoni, unknown, garden fresh y bbq chicken bacon, 4 clases, 5000 max_step con 200 imágenes de entrenamiento.

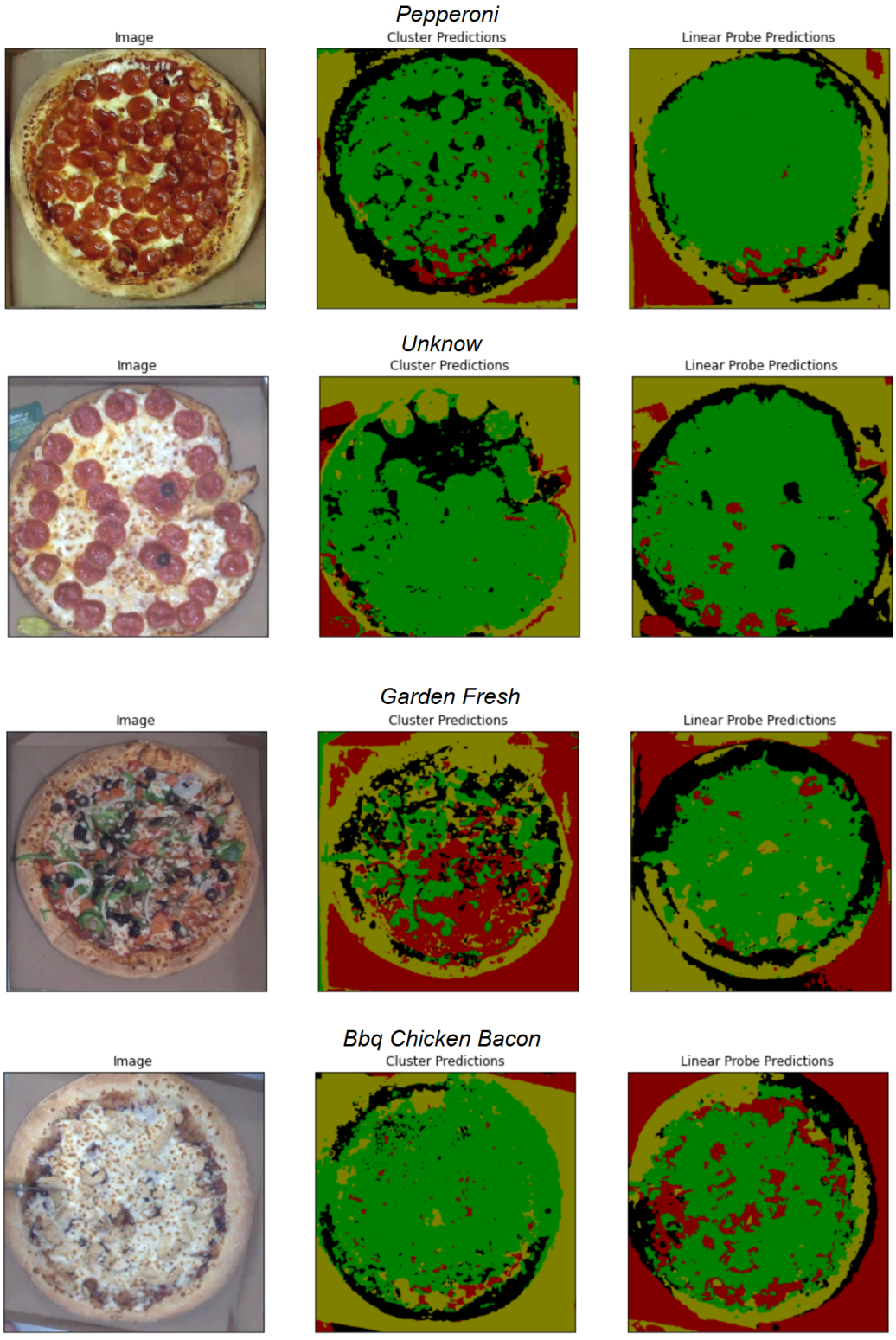


Figura A.5: Visualización de resultados de STEGO sobre pizzas pepperoni, unknown, garden fresh y bbq chicken bacon, 4 clases, 15000 max_step con 400 imágenes de entrenamiento.

Anexo B. Resultados Visuales, Reentrenamiento de DeepLabv3

(a) Original (b) Segmentación (c) Máscara

Figura B.1: DeepLabv3 entrenado con 23 imágenes y 150 épocas para segmentar el exceso de queso, prueba sobre pizza de meatball pepperoni.

Anexo C. Modelo Multiclase vs Modelos Únicos:

Tabla C.1: Resultados 11 modelos especializados en 1 clase cada uno. Lr fijo en 1e-4. Batch fijo en tamaño 4.

Ingrediente	ACC	RECALL	PRECISION	SPECIFITY	F1	mIoU
Excess	0.969	0.45	0.372	0.981	0.407	0.256
Right	0.885	0.887	0.789	0.885	0.835	0.717
Low	0.965	0.383	0.418	0.983	0.4	0.25
Meat	0.988	0.417	0.387	0.994	0.401	0.251
Olive	0.995	0.452	0.516	0.998	0.482	0.318
Chicken	0.993	0.46	0.536	0.997	0.495	0.329
Meatball	0.951	0.83	0.17	0.953	0.282	0.165
Green_Veg	0.987	0.739	0.537	0.991	0.622	0.451
Tomato	0.986	0.52	0.57	0.994	0.544	0.374
Crust	0.948	0.876	0.801	0.961	0.837	0.719
Pepp	0.972	0.835	0.799	0.983	0.817	0.69
PROM	**0.967**	**0.623**	**0.536**	**0.975**	**0.557**	**0.411**

Tabla C.2: Comparación resultados mejor modelo unificado (lr=1e-4 y batch=6) vs 11 modelos especializados en 1 clase cada uno, (lr = 1e-4 y batch = 4).

Ingrediente	ACC	RECALL	PRECISION	SPECIFITY	F1	mIoU
Excess_1	0.969	0.450	0.372	0.981	0.407	0.256
Excess_M	0.966	0.483	0.350	0.978	0.406	0.255
Right_1	0.885	0.887	0.789	0.885	0.835	0.717
Right_M	0.881	0.739	0.877	0.950	0.802	0.669
Low_1	0.965	0.383	0.418	0.983	0.400	0.25
Low_M	0.968	0.437	0.473	0.985	0.454	0.293
Meat_1	0.988	0.417	0.387	0.994	0.401	0.251
Meat_M	0.986	0.392	0.319	0.992	0.352	0.213
Olive_1	0.995	0.452	0.516	0.998	0.482	0.318
Olive_M	0.994	0.566	0.462	0.997	0.509	0.341
Chicken_1	0.993	0.46	0.536	0.997	0.495	0.329
Chicken_M	0.993	0.42	0.551	0.990	0.477	0.313
Meatball_1	0.951	0.83	0.17	0.953	0.282	0.165
Meatball_M	0.994	0.71	0.775	0.998	0.741	0.589
Green_Veg_1	0.987	0.739	0.537	0.991	0.622	0.451
Green_Veg_M	0.988	0.734	0.568	0.992	0.640	0.471
Tomato_1	0.986	0.520	0.570	0.994	0.544	0.374
Tomato_M	0.986	0.587	0.569	0.993	0.578	0.407
Crust_1	0.948	0.876	0.801	0.961	0.837	0.719
Crust_M	0.950	0.836	0.836	0.971	0.836	0.719
Pepp_1	0.972	0.835	0.799	0.983	0.817	0.69
Pepp_M	0.973	0.805	0.829	0.987	0.817	0.69
PROM_1	**0.967**	**0.623**	**0.536**	**0.975**	**0.557**	**0.411**
PROM_M	**0.972**	**0.610**	**0.601**	**0.986**	**0.601**	**0.451**