CAPÍTULO 2

EVOLUÇÃO E ARQUITETURA DA FAMÍLIA YOLO

1. CONTEXTO HISTÓRICO DA VISÃO COMPUTACIONAL: DO INVERNO DA IA À REVOLUÇÃO DO DEEP LEARNING

1.1 Os Primórdios das Redes Convolucionais e as Limitações da Era Inicial

Nas primeiras décadas de desenvolvimento da Inteligência Artificial, o treinamento de redes neurais artificiais com múltiplas camadas profundas enfrentava severas barreiras matemáticas e computacionais. O ajuste iterativo dos parâmetros por meio do algoritmo de retropropagação do erro (backpropagation) demonstrava forte instabilidade em arquiteturas complexas, frequentemente colapsando devido ao fenômeno do desvanecimento do gradiente (vanishing gradient). Nesse processo, os sinais de correção calculados a partir da camada de saída tornavam-se infinitesimalmente pequenos à medida que retropropagavam pelas camadas iniciais, impedindo o aprendizado efetivo. Somada à extrema escassez de dados anotados e ao reduzido poder de processamento dos computadores da época, essa limitação conduziu a comunidade científica a um período de relativo ceticismo e estagnação de investimentos conhecido como o “Inverno da IA” (AI Winter). Contudo, as Redes Neurais Convolucionais (Convolutional Neural Networks – CNNs) não surgiram subitamente com as grandes competições da década de 2010; seus fundamentos teóricos e estruturais foram estabelecidos ainda na década de 1980. Inspirado nas descobertas neurofisiológicas sobre a organização do córtex visual biológico de mamíferos (HUBEL; WIESEL, 1959, 1962), Fukushima (1980) introduziu a arquitetura do Neocognitron. Esse modelo pioneiro formalizou os conceitos de campos receptivos locais — nos quais cada unidade de processamento analisa apenas um fragmento espacial restrito da imagem de entrada — e de compartilhamento de pesos entre neurônios, reduzindo drasticamente a quantidade de parâmetros a serem ajustados em comparação com redes completamente conectadas. O salto prático subsequente ocorreu em 1989, quando Yann LeCun e seus colaboradores demonstraram de forma inédita que o algoritmo de backpropagation podia ser utilizado para aprender autonomamente os coeficientes dos filtros convolucionais diretamente a partir dos pixels brutos de imagens de dígitos manuscritos (LECUN; BOSER et al., 1989). Essa contribuição evoluiu na década de 1990 para o desenvolvimento da famosa arquitetura LeNet-5 (LECUN; BOTTOU et al., 1998), que obteve sucesso em aplicações comerciais de automação bancária e de triagem postal nos Estados Unidos, realizando a leitura automatizada de códigos de endereçamento (ZIP codes) e numerações de cheques. A despeito dessa demonstração bem-sucedida em tarefas de domínio restrito, a abordagem baseada em convoluções estagnou temporariamente ao final dos anos 1990. O grande obstáculo da época residia na impossibilidade de escalar a metodologia para a percepção de imagens naturais complexas e de alta resolução. A capacidade de processamento das unidades centrais de processamento (CPUs) existentes e a ausência de grandes acervos

de dados anotados inviabilizavam o treinamento de redes convolucionais com capacidade suficiente para atuar no mundo real.

1.2 O Desafio da Visão Real e o Paradigma da Engenharia Manual de Atributos

Diante da inviabilidade computacional de treinar redes neurais profundas ao longo dos anos 2000, a pesquisa em Visão Computacional voltou-se para a busca de métodos alternativos capazes de responder à indagação central do domínio: como reconhecer e localizar categorias genéricas de objetos em imagens reais do mundo cotidiano? Para contornar as limitações de aprendizado direto da imagem bruta, estabeleceu-se o paradigma da engenharia manual de características visuais (handcrafted feature engineering). Nessa abordagem tradicional, a interpretação visual dependia obrigatoriamente da intervenção de especialistas humanos, que projetavam matematicamente algoritmos dedicados à extração de descritores invariantes a mudanças de iluminação, escala e orientação. O fluxo operacional dividia-se rigidamente em duas etapas desconectadas: a imagem de entrada era submetida a algoritmos de extração manual de atributos — tais como o Scale-Invariant Feature Transform (SIFT), o Histogram of Oriented Gradients (HOG), o Local Binary Patterns (LBP) ou a representação por Fisher Vectors — e os vetores numéricos resultantes eram fornecidos a um classificador estatístico independente, tipicamente uma Máquina de Vetor de Suporte (Support Vector Machines – SVM). O avanço desse paradigma foi impulsionado pelo estabelecimento do desafio PASCAL VOC (Visual Object Classes Challenge) a partir de 2005 (EVERINGHAM et al., 2010). O PASCAL VOC desempenhou um papel divisor ao consolidar um benchmark padronizado para a comunidade internacional, integrando um acervo de imagens rotuladas, métricas formais de avaliação — como a Precisão Média (mean Average Precision – mAP) — e uma competição científica anual para tarefas de classificação, localização e detecção de objetos. Entretanto, como a extração de atributos dependia da intuição e do projeto humano, os sistemas baseados em descritores manuais encontravam dificuldades para generalizar o aprendizado quando submetidos a variações de oclusão, fundo dinâmico e deformação morfológica dos objetos.

1.3 A Infraestrutura do Big Data Visual: ImageNet e ILSVRC

À medida que a capacidade computacional evoluía no final da década de 2000, a escassez de dados padronizados em grande escala manteve-se como o gargalo principal para o avanço dos modelos de aprendizado. Em 2009, uma equipe de pesquisadores liderada pela professora Fei-Fei Li apresentou o projeto ImageNet (DENG et al., 2009), propondo a criação de um acervo visual massivo estruturado de acordo com a hierarquia taxonômica do banco léxico WordNet. O ImageNet superou as dimensões de todos os conjuntos de dados pré-existentes, reunindo milhões de imagens cuidadosamente rotuladas em milhares de categorias conceituais. Com base nessa infraestrutura de dados, foi instituído em 2010 o ImageNet Large Scale Visual Recognition Challenge (ILSVRC), que se consagrou como a competição de referência global na área de Visão Computacional.

O ILSVRC estabeleceu um ambiente competitivo anual em escala inédita, avaliando algoritmos sob mais de 1.000 categorias distintas de objetos e ultrapassando 1,4 milhão de imagens de treinamento. O desafio estruturava-se em três modalidades principais de avaliação: a classificação global da imagem, a localização da caixa delimitadora do objeto principal e a detecção de múltiplos objetos presentes na cena.

1.4 A Ruptura Tecnológica da AlexNet em 2012

Até o ano de 2011, as edições do ILSVRC eram amplamente dominadas por modelos baseados na evolução incremental da engenharia de características tradicionais, combinando descritores SIFT refinados, codificação por Fisher Vectors e classificadores SVM. Embora precisos para a época, os ganhos anuais de desempenho tornavam-se progressivamente menores, indicando a saturação dos métodos de extração manual. A ruptura definitiva desse cenário ocorreu no desafio ILSVRC de 2012, quando uma equipe da Universidade de Toronto — integrada por Alex Krizhevsky, Ilya Sutskever e Geoffrey Hinton — inscreveu uma Rede Neural Convolucional profunda de oito camadas, denominada AlexNet (KRIZHEVSKY; SUTSKEVER; HINTON, 2012). Os resultados da competição revelaram um salto de desempenho sem precedentes na história do processamento de imagens: enquanto o segundo colocado no desafio — utilizando as técnicas mais avançadas de engenharia manual de atributos — obteve uma taxa de erro top-5 de aproximadamente 26,2%, a AlexNet alcançou a marca de 15,3%. A margem de superação de 10,8 pontos percentuais não representou uma simples otimização incremental, mas a demonstração empírica da superioridade do aprendizado profundo de representações sobre os descritores manuais.

1.5 Fundamentos da AlexNet e a Transição Conceitual para o Aprendizado End-to-End

O feito extraordinário da AlexNet viabilizou-se pela convergência síncrona de três elementos indispensáveis: a formulação de uma arquitetura convolucional profunda com cerca de 60 milhões de parâmetros, o volume massivo de amostragem fornecido pelo ImageNet e o emprego pioneiro de unidades de processamento gráfico (GPUs NVIDIA) programadas via CUDA. A aceleração por GPU permitiu realizar o treinamento das operações matriciais em poucos dias, tarefa que exigiria meses em processadores convencionais (CPUs). Mais do que o ganho estatístico, a AlexNet promoveu uma mudança conceitual profunda no desenvolvimento da inteligência visual. No paradigma anterior a 2012, o fluxo analítico dependia da seleção prévia de características pelo pesquisador, seguida pelo treinamento isolado do classificador. Com o advento das redes convolucionais profundas, o processo passou a ser operado de ponta a ponta (end-to-end), cabendo à própria rede extrair automaticamente as representações relevantes diretamente da matriz de pixels da imagem. Esse aprendizado ocorre de maneira hierárquica ao longo da profundidade da rede: nas camadas convolucionais iniciais, os filtros aprendem de forma autônoma a

detectar primitivas visuais simples, tais como bordas direcionadas, gradientes de cor e texturas locais; nas camadas intermediárias, essas primitivas são combinadas para compor geometrias, contornos e partes reconhecíveis de objetos; e nas camadas finais e mais profundas, a rede sintetiza estruturas semânticas complexas e altamente abstratas, como rostos, estruturas corporais, componentes mecânicos ou silhuetas de animais. Dessa forma, a necessidade de especificação manual de atributos visuais foi definitivamente eliminada.

1.6 A Linha do Tempo da Visão Computacional e a Evolução rumo à Detecção em Tempo Real

Embora o impacto inicial da AlexNet tenha se tornado célebre na classificação de imagens, a competição ILSVRC contemplava de forma integrada o desafio de localização e detecção de objetos. A retrospectiva oficial da competição evidencia que 2012 marcou o ponto de inflexão a partir do qual as redes neurais profundas foram expandidas da mera atribuição de rótulos globais para a delimitação espacial de múltiplos objetos em cenas complexas. Essa evolução histórica estende-se em um contínuo que interliga marcos teóricos e práticos: iniciando na proposição dos campos receptivos do Neocognitron por Fukushima em 1980; passando pelas primeiras aplicações de backpropagation convolucional por LeCun em 1989 e 1998; avançando pela padronização de métricas do PASCAL VOC em 2005 e pela estruturação do ImageNet em 2009 e ILSVRC em 2010; perpassando o ápice dos métodos manuais em 2011 e a ruptura da AlexNet em 2012; culminando no surgimento das arquiteturas de detecção por regiões de interesse, como a R-CNN de Girshick et al. (2014) em 2014, e na introdução das redes residuais profundas (ResNet) por He et al. (2016) em 2015. Esse percurso culminou na reformulação das três indagações fundamentais que acompanharam a história da disciplina:

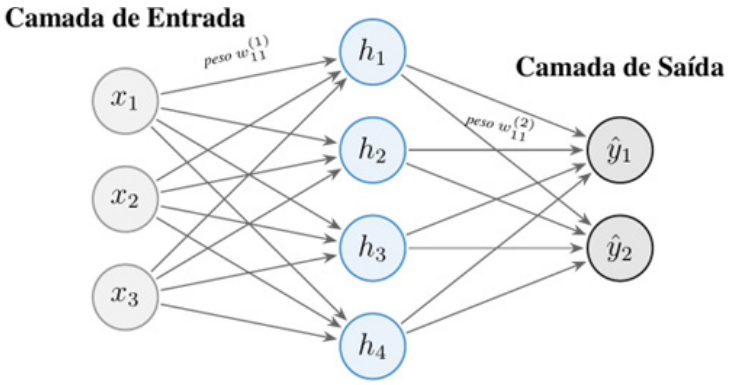
1. Na Era Tradicional (pré-2012), a questão central era: como projetar manualmente uma função matemática capaz de descrever as características de um objeto?
2. Na Era do Aprendizado Profundo em dois estágios (2012–2015), a indagação evoluiu para: como fazer a rede neural aprender autonomamente a extrair atributos para localizar e classificar regiões de interesse na imagem?
3. Na Era do Processamento em Tempo Real (2016–presente), a busca concentrou-se em: como realizar a extração de características, a localização espacial e a classificação categórica simultaneamente em uma única passagem direta, atingindo velocidade suficiente para operação em tempo real?

É precisamente para responder a essa terceira questão que a família de arquiteturas YOLO (You Only Look Once), concebida por Redmon, Divvala et al. (2016), foi desenvolvida e passou por sucessivas evoluções tecnológicas, conforme detalhado nas seções a seguir.

2. O QUE É UMA REDE NEURAL ARTIFICIAL: ESTRUTURA, GRAFOS, PESOS E APRENDIZADO

Para compreender a fundo o funcionamento de redes convolucionais complexas como as da família YOLO, é fundamental estabelecer primeiramente o conceito básico de uma **Rede Neural Artificial** (Artificial Neural Network – ANN). Uma Rede Neural Artificial é um modelo matemático e computacional inspirado na estrutura biológica do cérebro humano, projetado para reconhecer padrões, aprender relações complexas e realizar inferências a partir de dados de entrada. Do ponto de vista da teoria dos grafos, uma rede neural é representada como um **grafo direcionado ponderado**, composto por vértices (denominados neurônios ou nós) e arestas direcionadas que conectam esses neurônios em uma topologia organizada por camadas. A Figura 11 ilustra esquematicamente a arquitetura em grafo de uma rede neural artificial simples do tipo Perceptron Multicamadas (Multilayer Perceptron – MLP).

Figura 11: Representação de uma Rede Neural Artificial como um grafo direcionado ponderado: a informação flui da Camada de Entrada através da Camada Oculta até a Camada de Saída pelas arestas associadas a pesos computacionais (w_{ij}).



2.1 Anatomia do Grafo Neural: Camadas, Arestas, Pesos e Viéses

Conforme apresentado no grafo da Figura 11, a arquitetura de uma rede neural é constituída por três elementos matemáticos primordiais:

1. **Camadas (Layers):**
   - Camada de Entrada (Input Layer): É a primeira camada do grafo, responsável por receber os valores numéricos brutos não processados (x_1, x_2, . . . , x_n), tais como a intensidade de pixels de uma imagem ou características quantitativas.
   - Camadas Ocultas (Hidden Layers): São as camadas intermediárias localizadas entre a entrada e a saída. Nessas camadas, cada neurônio combina as informações recebidas das arestas anteriores e aplica transformações não lineares, permitindo extrair representações abstratas e padrões de ordem superior a partir dos dados.
   - Camada de Saída (Output Layer): É a camada final que entrega as respostas

ou predições calculadas pelo modelo ($\hat{y}_1$, $\hat{y}_2$), tais como as probabilidades de pertencimento a uma determinada classe categórica ou as coordenadas de uma caixa delimitadora.

2. Arestas Conectivas e Pesos (Weights – w_{ij}): As arestas são os canais direcionados que conectam os neurônios de uma camada aos neurônios da camada subsequente. Cada aresta possui associado a si um parâmetro numérico escalar denominado **peso** (w_{ij}). O peso quantifica a intensidade, força ou grau de relevância da conexão entre o neurônio ***i*** da camada anterior e o neurônio ***j*** da camada posterior. Se o peso for um valor positivo elevado, o sinal de entrada é fortemente amplificado; se for próximo de zero, o sinal é atenuado ou desconsiderado; se for negativo, a conexão exerce efeito inibitório.

3. Viéses (Biases – b_j): Além dos pesos, cada neurônio das camadas ocultas e de saída possui um termo ajustável independente denominado **viés** (b). O viés atua como uma constante de deslocamento que permite alterar o limiar de disparo do neurônio, garantindo flexibilidade para que ele consiga ser ativado mesmo quando todas as entradas numéricas resultarem em zero. Matematicamente, cada neurônio ***j*** em uma camada oculta calcula primeiramente o somatório ponderado de suas entradas somado ao seu viés:

$$z_j = \sum_{i=1}^{n} w_{ij} x_i + b_j \qquad (1)$$

Em seguida, o valor escalar resultante z_j é submetido a uma **função de ativação não linear** $\sigma(z)$ (como a função ReLU, Sigmoide ou Tanh), produzindo o sinal de saída do neurônio $a_j = \sigma(z_j)$ que será transmitido pelas arestas da camada seguinte.

2.2 Por que os Pesos Precisam ser Atualizados?

Quando uma rede neural é criada e inicializada, seus parâmetros internos — isto é, todos os pesos w_{ij} de suas arestas e os viéses b_j de seus neurônios — recebem valores numéricos aleatórios (ou pseudo-aleatórios pequenos). Consequentemente, durante as primeiras passagens do modelo, as respostas produzidas na camada de saída ($\hat{y}$) são completamente incorretas e distantes do gabarito real (y). A essência do **aprendizado de máquina** reside no processo iterativo de ajuste e calibração desses parâmetros. Os pesos precisam ser continuamente atualizados durante a fase de treinamento pelos seguintes motivos centrais:

- **Minimização do Erro de Predição:** Os pesos iniciais geram um erro elevado, quantificado matematicamente pela **Função de Perda** (L). Atualizar os pesos é o único meio de alterar a sensibilidade dos neurônios para fazer a resposta prevista $\hat{y}$ se aproximar do valor real y.
- **Aprendizado de Padrões e Generalização:** Cada peso ajustado reflete a importância atribuída a uma característica visual específica. À medida que os pesos são refinados

ao longo de múltiplas épocas de treinamento, a rede deixa de emitir palpites aleatórios e passa a "memorizar" a combinação ideal de atributos (como bordas, contornos e geometrias) necessária para classificar e localizar objetos em imagens inéditas.

- **Mecanismo da Retropropagação (Backpropagation):** A atualização é orientada pelo cálculo dos gradientes do erro em relação a cada peso ($\partial L/\partial w_{ij}$) via regra da cadeia. Um algoritmo de otimização (como o Gradiente Descendente / SGD ou Adam) modifica suavemente o valor de cada peso na direção oposta ao gradiente:

$$w_{ij}^{(novo)} = w_{ij}^{(antigo)} - \eta \cdot \frac{\partial L}{\partial w_{ij}} \qquad (2)$$

onde η representa a taxa de aprendizado (learning rate). Sem essa atualização sistemática dos pesos e viéses, a rede neural seria apenas um grafo estático sem capacidade adaptativa. É precisamente por meio da modificação progressiva desses pesos que a rede adquire inteligência computacional, permitindo que arquiteturas avançadas — como os modelos da família YOLO — aprendam a detectar objetos com elevada precisão em tempo real.

3. CONCEITO FUNDAMENTAL E A REVOLUÇÃO DO YOLO

O desenvolvimento da visão computacional moderna foi profundamente transformado pela introdução da arquitetura YOLO (You Only Look Once), concebida originalmente por Redmon, Divvala et al. (2016). Até o surgimento dessa abordagem, a tarefa de detecção de objetos em imagens dependia majoritariamente de sistemas em duas etapas (two-stage detectors), como a família R-CNN (Region-based Convolutional Neural Networks). Nesses modelos tradicionais, um primeiro módulo gerava propostas de regiões de interesse (region proposals) e, em seguida, uma rede convolucional dedicada classificava e ajustava as caixas de cada região proposta individualmente. Embora precisos, tais métodos apresentavam alto custo computacional e latência elevada, inviabilizando aplicações em tempo real. O YOLO revolucionou esse cenário ao formular a detecção de objetos como um único problema de regressão direta, operando de ponta a ponta (end-to-end). Em vez de segmentar a imagem em múltiplos sub-processos, uma única rede neural convolucional analisa a imagem inteira de uma só vez. Nessa passagem única, o modelo divide a imagem de entrada em uma grade (grid) e prevê simultaneamente as coordenadas das caixas delimitadoras (bounding boxes), as pontuações de confiança (confidence scores) e as probabilidades de cada classe para todos os objetos presentes (REDMON; DIVVALA et al., 2016). Essa arquitetura integrada proporcionou um ganho drástico de velocidade de processamento, permitindo que a detecção ocorresse em tempo real, característica que passou a definir e fundamentar toda a evolução da família YOLO nos anos seguintes.

4. TRAJETÓRIA EVOLUTIVA: DAS VERSÕES CLÁSSICAS ÀS ARQUITETURAS MODERNAS

Desde a publicação do trabalho original em 2016, a arquitetura YOLO passou por sucessivas reformulações impulsionadas tanto pela indústria quanto pela comunidade acadêmica. Neste capítulo analisamos a evolução histórica e seis das principais versões representativas do paradigma YOLO, compreendendo modelos amplamente utilizados em tarefas de visão computacional, agricultura e monitoramento de fauna.

4.1 Resumo da Evolução Clássica: YOLOv1 a YOLOv7

Antes de explorar as versões mais recentes (do YOLOv8 ao YOLO26), é essencial compreender os marcos teóricos estabelecidos pelas primeiras sete gerações da arquitetura:

- **YOLOv1** (REDMON; DIVVALA et al., 2016): Formulou a detecção como problema único de regressão direta em grade $S \times S$, dividindo a imagem em células e prevendo caixas e pontuações de confiança simultaneamente.
- **YOLOv2 (YOLO9000)** (REDMON; FARHADI, 2017): Introduziu caixas âncora (*anchor boxes*) pré-calculadas por agrupamento K-means, normalização por lote (*batch normalization*) em todas as camadas e resolução variável.
- **YOLOv3** (REDMON; FARHADI, 2018): Adotou a espinha dorsal Darknet-53 e predição multiescala em três níveis de resolução distintos (com conexões no estilo FPN), aprimorando substancialmente a detecção de objetos pequenos.
- **YOLOv4** (BOCHKOVSKIY; WANG; LIAO, 2020): Estruturou a arquitetura CSPDarknet53 e introduziu o neck PANet, além de otimizações do tipo *Bag of Freebies* (como aumento por Mosaico) e *Bag of Specials* (como blocos SAM e Mish).
- **YOLOv5** (JOCHER; CHAURASIA; QIU, 2023): Implementado nativamente em PyTorch pela Ultralytics, popularizou o ecossistema amigável de treinamento, validação e exportação automatizada para produção.
- **YOLOv6 & YOLOv7** (WANG; BOCHKOVSKIY; LIAO, 2023): Introduziram a reparametrização estrutural de módulos no treinamento (RepVGG) e o bloco E-ELAN, elevando o desempenho e reduzindo os custos de inferência. A partir do YOLOv8, o paradigma evoluiu de arquiteturas baseadas em âncoras (*anchor-based*) para modelos *anchor-free*, conforme detalhado a seguir.

4.2 YOLOv8: Consolidação Generalista e Multitarefa

Lançado pela Ultralytics em 2023, o YOLOv8 consolidou-se como um divisor de águas na família ao transformar o modelo em uma plataforma unificada e generalista para tarefas visuais (ULTRALYTICS, 2023). A arquitetura do YOLOv8 foi projetada sem o uso de âncoras pré-definidas (anchor-free), simplificando o processo de predição de caixas delimitadoras e aumentando a capacidade de generalização em cenários complexos. Além da detecção tradicional de objetos, o YOLOv8 introduziu suporte nativo e otimizado no

mesmo ambiente para:

- **Classificação de imagens:** atribuição de rótulos globais à cena visual;
- **Segmentação de instâncias:** delineamento de máscaras de pixels para cada objeto individual;
- **Estimativa de pose (pose estimation):** identificação de pontos de articulação e esqueletos corporais em humanos ou animais. Devido a essa versatilidade e à facilidade de integração oferecida por sua API, o YOLOv8 tornou-se uma das ferramentas mais populares em pesquisas aplicadas à fauna e à conservação biológica. Na literatura científica recente, o modelo surge tanto como arquitetura principal para contagem e monitoramento ecológico quanto como baseline de comparação de referência para a avaliação de novas versões (ZHU, Y. et al., 2025; RAHMAN; GIORDANO; PAGANO, 2025).

4.3 YOLOv9 e YOLOv10: Avanços Acadêmicos e Otimizações de Fluxo

Enquanto os desenvolvimentos comerciais focavam em usabilidade, grupos acadêmicos propuseram inovações estruturais profundas no fluxo de informação e na eficiência dos modelos:

1. YOLOv9 (Wang, Yeh & Liao, 2024): Desenvolvido por Wang, Yeh e Liao (2024), o YOLOv9 atacou o problema do "gargalo de informação"(information bottleneck) inerente a redes neurais profundas, onde dados visuais relevantes são gradualmente perdidos à medida que transitam pelas camadas da rede. Para mitigar esse efeito, os autores introduziram:

- **Programmable Gradient Information (PGI):** uma estrutura de gradiente programável que garante a preservação de atributos essenciais ao longo do treinamento;
- **Generalized Efficient Layer Aggregation Network (GELAN):** uma arquitetura leve e flexível de agregação de camadas que otimiza o uso de parâmetros e recursos computacionais em diferentes dimensões de rede. Essa capacidade de retenção de características demonstrou grande valor no monitoramento de espécies em habitats naturais complexos, como na identificação de antílopes-tibetanos (Pantholops hodgsonii) em armadilhas fotográficas (ZHU, R. et al., 2024). 2. YOLOv10 (Wang et al., 2024): Apresentado por Ao Wang et al. (2024), o YOLOv10 concentrou seus avanços na redução da latência de inferência através de uma abordagem end-to-end real sem pós-processamento. Tradicionalmente, os modelos YOLO dependiam da técnica de Supressão de Não Máximos (Non-Maximum Suppression – NMS) para eliminar caixas delimitadoras duplicadas e sobrepostas na etapa final de inferência. O YOLOv10 dispensou a necessidade do NMS ao adotar uma estratégia de atribuição dupla de rótulos consistentemente pareada durante o treinamento, garantindo que o modelo preveja apenas uma caixa válida por objeto. Essa otimização reduziu drasticamente o

tempo total de inferência, permitindo sua aplicação prática no MegaDetector v6 para filtrar volumes massivos de imagens vazias obtidas em armadilhas fotográficas no campo (ZHANG et al., 2026).

4.4 YOLO11: Eficiência Computacional e Extração de Atributos

Em 2024, a Ultralytics disponibilizou o YOLO11 como a atualização direta da linha principal da família (ULTRALYTICS, 2024). As inovações do YOLO11 concentraram-se na reformulação dos blocos de extração de características no backbone e no neck da rede, melhorando a representação espacial e a eficiência no uso de parâmetros sem alterar o suporte multitarefa estabelecido no YOLOv8. Estudos recentes aplicados à fauna comparam frequentemente o YOLO11 com o YOLOv8 para mensurar os ganhos empíricos. Por exemplo, Corputty et al. (2024) registraram um desempenho superior do YOLO11 no reconhecimento de 14 classes distintas de animais. Da mesma forma, X. Wang et al. (2025) e Du et al. (2024) selecionaram a arquitetura YOLO11 como espinha dorsal para o desenvolvimento de sistemas automatizados de detecção de avifauna e mapeamento de espécies sob proteção ambiental.

4.5 YOLOv12: Incorporação de Mecanismos de Atenção em Tempo Real

Proposto por Tian, Ye e Doermann (2025), o YOLOv12 representou uma mudança de paradigma arquitetural ao integrar mecanismos de atenção (attention mechanisms) ao backbone do modelo. Historicamente, módulos baseados em atenção (como os Transformers) eram considerados computacionalmente caros e lentos para processamento visual em tempo real. O YOLOv12 superou essa limitação ao redesenhar os blocos de atenção para operar de maneira altamente paralelizada e otimizada, combinando a capacidade de modelagem de contexto global da atenção com a velocidade operacional tradicional das redes convolucionais. Contudo, avaliações experimentais no domínio ecológico indicam que aumentos na complexidade arquitetural nem sempre se traduzem em ganhos expressivos em todas as cenários de teste. Em tarefas de monitoramento animal, Y. Zhu et al. (2025) relataram um mAP@0.50 (mean Average Precision) de 92,2% para o YOLOv12, resultado ligeiramente superior ao mAP de 91,3% obtido pelo YOLOv8 no mesmo conjunto de dados. Além disso, pesquisas como a de Cai et al. (2025) avaliaram a versão compacta YOLOv12n em comparação direta com o YOLO11n e o YOLOv8n em dispositivos computacionais embarcados de baixo consumo.

4.6 YOLO26: Inferência End-to-End e Estratégias Avançadas de Treinamento

A iteração mais recente analisada neste capítulo é o YOLO26, desenvolvido por Jocher et al. (2026) como uma nova geração de modelos projetada para maximizar a eficiência computacional em uma ampla variedade de hardwares. Seguindo a premissa de eliminação de gargalos introduzida pelo YOLOv10, o YOLO26 realiza inferência nativa de ponta a ponta (end-to-end) sem NMS na sua configuração padrão. Entre as principais

inovações técnicas introduzidas pelo YOLO26, destacam-se:

- **Remoção da Distribution Focal Loss (DFL):** simplificação do cabeçote de detecção (detection head), reduzindo o overhead matemático na regressão de caixas;
- **Otimizador MuSGD:** novo algoritmo de otimização de gradientes projetado para estabilizar e acelerar a convergência durante o treinamento;
- **Função de Perda Progressiva (Progressive Loss):** ajuste dinâmico da ponderação dos erros ao longo das épocas de treinamento;
- **Atribuição de Rótulos Atenta a Objetos Pequenos (Small-Target-Aware Label Assignment – STAL):** estratégia de pareamento de amostras especificamente desenhada para aprimorar a detecção de objetos de pequenas dimensões na cena visual (JOCHER et al., 2026). Por ser uma contribuição científica de lançamento recente, o YOLO26 ainda representa uma fronteira em consolidação, com poucos trabalhos publicados reportando sua aplicação empírica na literatura de monitoramento da fauna.

5. ANÁLISE COMPARATIVA E COMPROMISSOS DE DESEMPENHO (TRADEOFFS)

A evolução da família YOLO reflete uma constante busca pelo equilíbrio ideal entre acurácia de detecção, velocidade de inferência (quadros por segundo – FPS) e viabilidade de implantação em hardware limitado. A escolha da versão mais adequada para um problema de visão computacional depende criticamente das restrições operacionais da aplicação. Em estudos aplicados ao monitoramento da vida selvagem, essas nuances ficam evidentes. Por exemplo, Adhikari et al. (2026) conduziram um benchmark comparativo abrangendo 12 variantes distintas pertencentes às gerações do YOLOv8 ao YOLO11 na detecção automatizada de cervos em vida livre. Os autores demonstraram que o tempo de inferência final é altamente dependente da interação entre a arquitetura do modelo e o hardware de execução (como GPUs de desktop vs. Processadores embarcados). Em consonância, Barros et al. (2024) chegaram a conclusões semelhantes ao comparar iterativamente as gerações do YOLOv8 ao YOLOv12 no reconhecimento e classificação de jabutis, ressaltando que modelos mais complexos nem sempre oferecem vantagens justificáveis em relação ao custo computacional adicional exigido. Assim, a análise das versões do YOLO estudadas fornece um panorama claro da maturidade e diversidade dos modelos de detecção em tempo real, permitindo a seleção fundamentada de arquiteturas segundo as necessidades de acurácia, tempo de resposta e recursos computacionais disponíveis.

6. PIPELINE DE TREINAMENTO, VALIDAÇÃO E TESTE DA ARQUITETURA YOLO

Para a implantação prática e o sucesso de modelos de detecção de objetos baseados no paradigma YOLO em cenários reais — como no monitoramento agrícola e faunístico —, é indispensável compreender como a arquitetura interna do modelo processa a informação

visual e como se estrutura metodologicamente o ciclo de treinamento, validação e teste.

6.1 Componentes Estruturais Internos do YOLO: Backbone, Neck e Head

A arquitetura dos modelos da família YOLO é organizada modularmente em três componentes principais encadeados, cada qual desempenhando uma função matemática e computacional específica no pipeline de detecção: 1. Backbone (Espinha Dorsal): É o módulo de extração primária de características visuais. O Backbone recebe a matriz tridimensional da imagem de entrada e, por meio de sucessivas camadas convolucionais, operações de redução espacial (pooling ou convoluções com stride) e blocos residuais, extrai mapas de atributos (feature maps) em múltiplos níveis de resolução. As camadas iniciais do Backbone capturam detalhes espaciais finos (bordas e texturas), enquanto as camadas mais profundas sintetizam informações semânticas abstratas. Variantes modernas do YOLO utilizam backbones altamente eficientes, tais como CSPDarknet no YOLOv8, GELAN no YOLOv9 e uma derivada aprimorada da CSPDarknet com blocos C3k2 e C2PSA no YOLO11 (enquanto a arquitetura HGNetv2 é utilizada no RT-DETR). 2. Neck (Pescoço): Atua como a estrutura de fusão multiescala de atributos localizada entre Backbone e o Head. A função principal do Neck é combinar mapas de características provenientes de diferentes profundidades do Backbone, mesclando informações de alta resolução espacial (essenciais para detectar objetos pequenos) com informações de alta riqueza semântica (essenciais para reconhecer objetos grandes). Para isso, o Neck emprega estruturas topológicas compostas por redes de pirâmides de atributos, tais como FPN (Feature Pyramid Network) e o PANet (Path Aggregation Network). Essa fusão garante que o modelo consiga detectar com elevada acurácia alvos de dimensões variadas presentes simultaneamente na mesma imagem. 3. Head (Cabeça de Detecção): É o módulo final de predição responsável por transformar os mapas de atributos refinados pelo Neck nas respostas de saída do modelo. O Head executa simultaneamente duas tarefas fundamentais: a regressão de caixas delimitadoras (calculando as coordenadas x, y, w, h de cada objeto) e a classificação categórica (atribuindo probabilidades e pontuações de confiança para cada categoria). Em versões ancoradas ou anchor-free, o Head produz as caixas finais que delimitam os alvos na cena visual.

6.2 Particionamento do Dataset e o Ciclo de Aprendizado

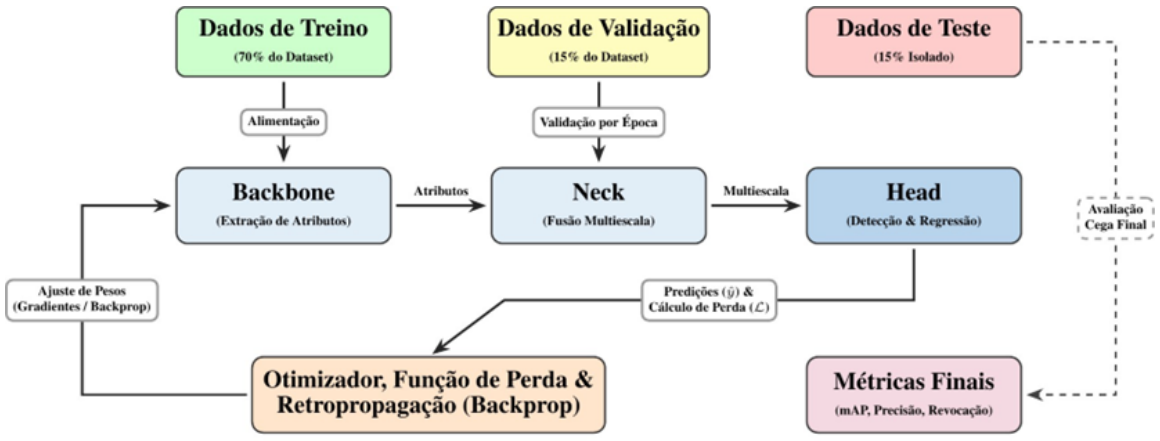
O desenvolvimento de um modelo YOLO exige a divisão metodológica do acervo de imagens anotadas em três conjuntos disjuntos e independentes, prevenindo o vazamento de dados (*data leakage*) e garantindo a capacidade de generalização do sistema:

- **Conjunto de Treinamento (Training Set):** Representa a maior fração do conjunto de dados (tipicamente entre 70% e 80%). Suas imagens e anotações são alimentadas diretamente no modelo durante a fase de aprendizado. Para cada lote de imagens, o Head gera predições que são comparadas com o gabarito real por meio de funções

de perda compostas — incluindo perdas de localização (CIoU ou GIoU), perdas de classificação (Binary Cross-Entropy) e perdas de distribuição. Os erros resultantes são retropropagados via *backpropagation* para atualizar iterativamente os pesos do Backbone, Neck e Head através de otimizadores como SGD, AdamW ou MuSGD.

- **Conjunto de Validação (Validation Set):** Compreende uma fração intermediária dos dados (tipicamente entre 10% e 15%). Ao final de cada época (*epoch*) de treinamento, o modelo com os pesos atuais é avaliado no conjunto de validação sem atualizar parâmetros. Essa etapa permite monitorar a evolução da Precisão Média (mAP@0.50 e mAP@0.50:0.95), ajustar hiperparâmetros (como taxa de aprendizado e tamanho de lote) e aplicar estratégias de parada prévia (*early stopping*) para evitar o superajuste (*overfitting*).
- **Conjunto de Teste (Test Set):** É uma parcela preservada e mantida totalmente isolada durante todo o processo de ajuste e treinamento (tipicamente entre 10% e 15%). O conjunto de teste é submetido ao modelo somente após a conclusão definitiva do treinamento para mensurar de forma imparcial e cega a acurácia, o tempo de inferência e a capacidade real de generalização da rede quando exposta a dados do mundo real. A Figura 12 sintetiza o fluxo de integração entre a arquitetura modular do YOLO (Backbone, Neck e Head), o particionamento do conjunto de dados e o ciclo iterativo de otimização de pesos.

Figura 12: Fluxo de informação e arquitetura modular do YOLO (Backbone, Neck e Head) integrado ao particionamento do dataset (Treino, Validação e Teste) e ao ciclo de otimização de pesos.



REFERÊNCIAS

ADHIKARI, B. et al. Evaluating Speed-Accuracy Trade-Offs of YOLOv8 and YOLO11 Variants for Deer Detection. Wildlife Society Bulletin, v. 50, e1345, 2026.

BARROS, Gustavo Moura et al. Avaliação comparativa de arquiteturas YOLO para identificação de jabutis e implementação do software TortoiseVision Detect. Revista de Derecho y Cambio Social, v. 23, e5475, 2024.

BOCHKOVSKIY, Alexey; WANG, Chien-Yao; LIAO, Hong-Yuan Mark.YOLOv4: Optimal

Speed and Accuracy of Object Detection. arXiv preprint arXiv:2004.10934, 2020. Disponível em: <https://arxiv.org/abs/2004.10934>.
CAI, H. et al. Comparative Analysis of Nano-Scale YOLO Variants (YOLOv8n, YOLO11n, and YOLOv12n) for Field Deployments. IEEE Access, v. 13, p. 14520–14535, 2025.
CORPUTTY, P. et al. Performance Benchmark of YOLO11 versus YOLOv8 across 14 Animal Classes. Ecological Indicators, v. 158, p. 111450, 2024.
DENG, Jia et al. ImageNet: A large-scale hierarchical image database. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR). [S.l.: s.n.], 2009.
P. 248–255. DU, Y. et al. Protected Animal Species Identification using Enhanced YOLO11 Backbone. Global Ecology and Conservation, v. 51, e02980, 2024.
EVERINGHAM, Mark et al. The PASCAL Visual Object Classes (VOC) Challenge. International Journal of Computer Vision, v. 88, n. 2, p. 303–338, 2010.
FUKUSHIMA, Kunihiko. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. Biological Cybernetics, v. 36, n. 4, p. 193–202, 1980.
GIRSHICK, Ross et al. Rich feature hierarchies for accurate object detection and semantic segmentation. In: PROCEEDINGS of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). [S.l.: s.n.], 2014. P. 580–587. HE, Kaiming et al. Deep residual learning for image recognition. In: PROCEEDINGS of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). [S.l.: s.n.], 2016. P. 770–778.
HUBEL, David H.; WIESEL, Torsten N. Receptive fields of single neurones in the cat's striate cortex. The Journal of Physiology, v. 148, n. 3, p. 574–591, 1959.
HUBEL, David H.; WIESEL, Torsten N. Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. The Journal of Physiology, v. 160, n. 1, p. 106–154, 1962.
JOCHER, Glenn et al. YOLO26: End-to-End Vision Model Architecture with Progressive Loss and Small-Target-Aware Assignment. Ultralytics Technical Report, 2026.
JOCHER, Glenn; CHAURASIA, Ayush; QIU, Jing. Ultralytics YOLO. [S.l.: s.n.], 2023. Software de detecção, segmentação, classificação e estimativa de pose baseado na família YOLO. Disponível em: <https://github.com/ultralytics/ultralytics>.
KRIZHEVSKY, Alex; SUTSKEVER, Ilya; HINTON, Geoffrey E. ImageNet classification with deep convolutional neural networks. In: ADVANCES in Neural Information Processing Systems (NeurIPS). [S.l.: s.n.], 2012. v. 25, p. 1097–1105.
LECUN, Yann; BOSER, Bernhard et al. Backpropagation applied to handwritten zip code recognition. Neural Computation, v. 1, n. 4, p. 541–551, 1989.
LECUN, Yann; BOTTOU, Léon et al. Gradient-based learning applied to document recognition. Proceedings of the IEEE, v. 86, n. 11, p. 2278–2324, 1998.
RAHMAN, M.; GIORDANO, M.; PAGANO, A. Deep Learning Architectures for Automated Fauna Detection: A Systematic Review. Computers and Electronics in Agriculture, v. 230, p. 109820, 2025.

REDMON, Joseph; DIVVALA, Santosh et al. You Only Look Once: Unified, Real-Time Object Detection. In: PROCEEDINGS of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). [S.l.: s.n.], 2016. P. 779–788.
REDMON, Joseph; FARHADI, Ali. YOLOv3: An Incremental Improvement. arXiv preprint arXiv:1804.02767, 2018. Disponível em: <https://arxiv.org/abs/1804.02767>.
TIAN, Y.; YE, Z.; DOERMANN, D. YOLOv12: Attention-Centric Real-Time Object Detection Framework. arXiv preprint arXiv:2502.12524, 2025. ULTRALYTICS. YOLO11: Next-Generation Real-Time Vision AI Framework. 2024. Disponível em: <https://github.com/ultralytics/ultralytics>. ULTRALYTICS. YOLOv8: State-of-the-Art Real-Time Object Detection and Instance Segmentation. 2023. Disponível em: <https://github.com/ultralytics/ultralytics>.
WANG, Ao et al. YOLOv10: Real-Time End-to-End Object Detection. arXiv preprint arXiv:2405.14458, 2024.
WANG, Chien-Yao; BOCHKOVSKIY, Alexey; LIAO, Hong-Yuan Mark. YOLOv7: Trainable Bag-of-Freebies Sets New State-of-the-Art for Real-Time Object Detectors. In: PROCEEDINGS of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. [S.l.: s.n.], 2023. P. 7464–7475. DOI: 10.1109/CVPR52729.2023.00721.
Disponível em: <https://doi.org/10.1109/CVPR52729.2023.00721>.
WANG, Chien-Yao; YEH, I-Hsuan; LIAO, Hong-Yuan Mark. YOLOv9: Learning What You Want to Learn Using Programmable Gradient Information. arXiv preprint arXiv:2402.13616, 2024.
WANG, X. et al. Automated Bird Species Detection and Counting using YOLO11 Architecture. Avian Research, v. 16, p. 100180, 2025.
ZHANG, L. et al. MegaDetector v6: Efficient Filtering of Empty Camera Trap Imagery Using NMS-Free YOLOv10. Remote Sensing in Ecology and Conservation, v. 12, e412, 2026.
ZHU, R. et al. Monitoring Tibetan Antelopes from Camera Trap Images Using YOLOv9 Architecture. Biological Conservation, v. 290, p. 110415, 2024.
ZHU, Y. et al. Comparative Evaluation of Object Detection Models in Wildlife Monitoring. Ecological Informatics, v. 80, p. 102510, 2025.